

IBM WebSphere Commerce



Administration Guide

Version 5.5

IBM WebSphere Commerce



Administration Guide

Version 5.5

Note:

Before using this information and the product it supports, be sure to read the information in “Notices” on page 171.

First Edition, Sixth Revision (March 2005)

This edition applies to IBM WebSphere Commerce Version 5.5 (product number 5724-A18) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality.

IBM welcomes your comments. You can send your comments by using the online IBM WebSphere Commerce documentation feedback form, available at the following URL:

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

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

© Copyright International Business Machines Corporation 2003. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	v	Viewing log files.	64
Where to find information.	v	Configuring LDAP	65
WebSphere Commerce publications.	v	Configuring rules services	66
WebSphere Commerce online help	v	Configuring the staging server	66
Further information on the Web.	vi	Configuring connectivity	71
I Summary of changes	vi	Configuring WebSphere Commerce Payments	71
Before you begin	vii		
Knowledge requirements	viii	Chapter 4. Hosted store administration 73	
Conventions used in this book.	viii	Administering resellers	73
Path variables.	ix	Administering distributors	73
		Site administration	74
Chapter 1. Introduction to administering			
WebSphere Commerce	1	Chapter 5. Dynamic caching	75
Administration tools.	2	WebSphere Commerce servlet or JSP result caching	75
Administration Console.	2	WebSphere Application Server dynamic cache	75
Configuration Manager.	2	Dynamic cache	75
WebSphere Commerce Payments	3	Enabling the dynamic cache service and servlet	
Organization Administration Console	9	caching.	76
WebSphere Commerce Accelerator	9	Enabling the Dynamic Cache Monitor	76
WebSphere Application Server	9	Tree View for Dependency IDs for WebSphere	
Database	10	Dynamic Cache Monitor	77
		Configuring cacheable objects	77
Chapter 2. Basic administration tasks 11		Caching defaults.	78
One-time-only administration tasks	11	Full page and fragment caching	79
Installation and configuration	11	Full page caching	79
Regular administration tasks.	11	Fragment cache	80
Starting and stopping WebSphere Commerce	11	Cache servlet filtering	80
Launching WebSphere Commerce Configuration		Store page caching strategy	82
Manager	12	Cache invalidation mechanisms in WebSphere	
Creating, configuring, and deleting an instance	14	Commerce.	82
Opening the Administration Console	16	Defining the invalidation policies in	
Opening the Organization Administration		cachespec.xml	83
Console.	17	Removing cache entries through the Cache	
Enabling and disabling WebSphere Commerce		Monitor	85
components	18	Invoking the dynamic cache invalidation APIs.	86
Starting and stopping WebSphere Commerce		Writing commands for command-based invalidation	86
Payments	18	An example of using command invalidation in	
Administrative tasks in the install guides	20	WebSphere Commerce.	86
		Cache invalidation example	87
Chapter 3. Configuration	21	Cache Invalidation API (DynaCacheInvalidation).	90
Configuration services.	21	CACHEIVL table	90
Architectural configuration	21	Invalidation in external caches	90
Messaging services	21	Web server plug-in cache.	90
Scheduling services.	21	Simple file servlet	91
Logging services.	22	Problem determination	92
WebSphere Commerce Payments Problem			
determination	25	Chapter 6. System maintenance	95
Rules services	25	Database Cleanup utility	95
LDAP — Lightweight Directory Access Protocol	26	Database Cleanup utility objects	96
Staging server	33	Database Cleanup commands	101
Connectivity services	57	Cleaning the database	105
Configuration tasks.	60	Setting the PATH environment variables for	
Configuring messaging services	60	WebSphere Commerce utilities	106
Configuring scheduling services	61	Deleting objects	106
Configuring logging	61	Deleting account objects.	106

Deleting address objects	107
Deleting available to promise inventory objects	107
Deleting attachment objects	108
Deleting auction objects	108
Deleting auction log objects	109
Deleting autobid log objects	109
Deleting base item objects	110
Deleting bid log objects	110
+ Deleting cacheivl objects	111
Deleting calculation code objects	111
Deleting campaign objects	112
Deleting campaign statistic objects	112
Deleting catalog entry objects	113
Deleting catalog group objects	113
Deleting contract objects	114
Deleting coupon objects	114
Deleting expected inventory record objects	115
Deleting details about expected inventory record objects	115
Deleting forum message objects	116
Deleting fulfillment center objects	116
Deleting inventory code adjustment objects	117
Deleting inventory adjustment objects	117
Deleting specified item information objects	118
Deleting member message relationship objects	118
Deleting message objects	119
Deleting order objects	119
Deleting organization objects	120
Deleting Product Advisor statistic objects	121
Deleting Product Comparison statistic objects	121
Deleting Product Explorer statistic objects	122
Deleting policy objects	122
Deleting product set objects	123
Deleting request-for-quote objects	123
Deleting returned item objects	124
Deleting return reason objects	124
Deleting Sales Assistant statistic objects	125
Deleting staged objects	125
Deleting store objects	126
Deleting user objects	126
Deleting user traffic log objects	127
Deleting vendor objects	127
Adding a new configuration to the Database	
Cleanup utility	128
Examples of deleting objects	129
Chapter 7. Performance	131
Performance monitoring using the WebSphere Commerce PMI module	132
Setting up the PMI module	132
Starting up Tivoli Performance Viewer	133
WebSphere Commerce PMI Module report values	134
Other performance tools	135
Maintaining the Scheduler	135
Replication for LDAP	136
Testing the site on a staging server	136

Configuring the staging server	136
Creating triggers for custom tables	137
Configuring a remote database	137
Copying data to the staging database	137
Running the Stage Check command	138
Propagating data to the production database	138
Copying files to the production server	139
Deleting staged objects	139
WebSphere Commerce Payments performance tuning parameters	140
Managing the Payments instance database on iSeries	142

Chapter 8. Troubleshooting 143

Adapter for CrossWorlds	143
WebSphere Commerce Payments	144
Tuning for high-performance environments	144

Appendix A. LDAP scenario 147

LDAP scenario: the LDAP server as the member repository	147
--	-----

Appendix B. LDAP files 149

ldapmap.dtd	149
ldapentry.xml	149

Appendix C. WebSphere Commerce Payments Tutorial 155

Step 1: Accessing the WebSphere Commerce Payments user interface	155
Step 2: Creating a WebSphere Commerce Payments merchant and authorizing a cassette	156
Step 3: Defining WebSphere Commerce Payments users	156
Step 4: Assigning user roles	157
Logging on as the Merchant Administrator	158
Step 5: Creating an account	158
Specifying account settings	159
Step 6: Creating a brand	160
Step 7: Creating orders using the Sample Checkout	160
Step 8: Approving orders	162
Approving orders from the Order page	162
Approving orders with the sale function	163
Step 9: Depositing payments	163
Step 10: Settling batches	164
Step 11: Issuing a credit	165
Step 12: Viewing batch totals	167

Appendix D. National Language Support (NLS) information for WebSphere Commerce Payments. . . 169

NLS hints and tips	169
Supported languages and locales	170

Notices 171

About this book

This book is intended for administrators and describes how to administer WebSphere® Commerce components using the Administration Console, and other administration tools.

Where to find information

WebSphere Commerce has online and hardcopy information describing the complete e-commerce solution. In addition, the software products that are bundled with WebSphere Commerce provide further information, describing the specific features and functions of the software. This section provides a quick overview of where to locate the various types of information.

WebSphere Commerce publications

- *WebSphere Commerce Fundamentals*, Version 5.5
- *WebSphere Commerce Programming Guide and Tutorials*, Version 5.5
- *WebSphere Commerce Installation Guide*, Version 5.5
- *WebSphere Commerce Quick Beginnings*, Version 5.5
- *WebSphere Commerce Additional Software Guide*, Version 5.5
- *WebSphere Commerce Migration Guide*, Version 5.5
- *WebSphere Commerce Store Development Guide*, Version 5.5
- *WebSphere Commerce Sample Store Guide*, Version 5.5
- *WebSphere Commerce Administration Guide*, Version 5.5
- *WebSphere Commerce Security Guide*, Version 5.5
- *WebSphere Commerce Payments Programming Guide and Reference*, Version 5.5
- *WebSphere Commerce Payments OfflineCard Cassette Supplement*, Version 5.5
- *WebSphere Commerce Payments CustomOffline Cassette Supplement*, Version 5.5

For updates to these publications, refer to the following Web addresses:

 Business Edition
(http://www.ibm.com/software/webservers/commerce/wc_be/)

 Professional Edition
(http://www.ibm.com/software/webservers/commerce/wc_pe/)

WebSphere Commerce online help

The WebSphere Commerce online help consists of online information that can be viewed using a Web browser.

The online help can be accessed from a Web browser that runs on Internet Explorer, Version 5.5, or higher using the following address:

`https://host_name:8000/wchelp`

where *host_name* is the name of your WebSphere Commerce machine.

Further information on the Web

WebSphere Commerce Support

To find support information, including newsgroups, FAQs, technical notes, troubleshooting information and downloads, visit the following Web addresses:

WebSphere Commerce Support Web page
(<http://www.ibm.com/software/commerce/support/>)

WebSphere Commerce Technical Library Web page
(<http://www.ibm.com/software/commerce/library/>)

Software partners

There are many software partners that offer products and services to enhance WebSphere Commerce. For information about these partners, visit the Commerce Zone (<http://www.ibm.com/software/wsdd/zones/commerce/>) and click the Software Developers link.

Redbooks™

To find more advanced technical information, visit the Redbooks Web site (<http://www.ibm.com/redbooks>) and search for WebSphere Commerce.

Summary of changes

This Guide and any updated versions of this Guide are available in the WebSphere Commerce Technical Library Web page (<http://www.ibm.com/software/commerce/library/>). For additional information for your WebSphere Commerce edition, see the overview pages:

- Business Edition
(http://www.ibm.com/software/webservers/commerce/wc_be/)
- Professional Edition
(http://www.ibm.com/software/webservers/commerce/wc_pe/)

For additional support information, see the WebSphere Commerce Support site (<http://www.ibm.com/software/commerce/support/>).

To learn about last-minute changes to the product, see the updated product README file, also available from the above Web site. For instructions on how to install WebSphere Commerce and its supported products, see the *WebSphere Commerce Installation Guide*.

Updates from the last version of this book are identified by revision characters contained in the margin. This book uses the following conventions for revision characters:

- The "+" (plus) character identifies updates that have been made in the current version of this book.
- The "|" (split vertical bar) character identifies updates that have been made in the previous versions of this book.

The following table shows the main changes that have been made to this book.

Change	Chapter or page affected
Corrected dbclean examples for the iSeries™ platform.	Chapter 6, "System maintenance," on page 95
Corrected dbclean and stagingcopy, stagingcheck, and stagingprop examples for the iSeries platform.	Chapter 7, "Performance," on page 131 and "Staging server" on page 33
The section "Logging for stand-alone tools" is no longer used and has consequently been removed.	"Configuring logging" on page 61
Clarification and miscellaneous updates to information on dynamic caching.	"Configuring cacheable objects" on page 77
Added Database Cleanup (dbclean) and Staging Server utility syntax for the Linux Hybrid configuration (when used with a DB2 for z/OS database)	"Database Cleanup utility" on page 95 and "Staging server commands" on page 35
Added note on caching consideration when using PVC adapter	Page 78
Updated Enabling Trace information	"Enabling tracing components" on page 61
Updated dbclean examples to include the -dbuser and -dbpasswd parameters.	"Database Cleanup utility" on page 95
Updated dbclean to change cachelog objects to chacheivl objects and update "lastsession" to "prevlastsession" in accordance with changes due to WebSphere Commerce fix pack version 5.5.06.	"Database Cleanup utility" on page 95

Before you begin

The IBM® WebSphere Commerce Administration Guide provides information about the tasks a site administrator or system administrator is required to perform in the WebSphere Commerce site. This guide refers to either role as a Site Administrator role.

Important

Throughout this book you will see platform-specific information. Ensure to follow the instructions specific to your platform.

This book is divided into the following chapters:

Introduction to administering WebSphere Commerce

Basic administration tasks

Configuration

Hosted store administration

Dynamic caching

System maintenance

Performance

Troubleshooting

Appendix A: LDAP scenario

Appendix B : LDAP files

Appendix C: WebSphere Commerce Payments Tutorial

Appendix D: National Language Support (NLS) information for WebSphere Commerce Payments

Knowledge requirements

This book should be read by site administrators or system administrators that need to understand how to set up and maintain a WebSphere Commerce site. Administrators must have knowledge in the following areas:

- E-commerce solution architecture
- Installation and configuration of network interface components such as, firewalls, DFS™, DNS, routers, and network hubs.
- Database technology.
- XML
- Store business procedures.

Conventions used in this book

This book uses the following conventions:


Boldface type indicates graphical user interface (GUI) controls such as names of fields, buttons, or menu choices.

Monospaced type indicates examples of text you enter exactly as shown, as well as directory paths.


Italic type is used for emphasis and variables for which you substitute your own values.



indicates additional information that can help you complete a task.

 indicates information specific to WebSphere Commerce for the IBM Eserver iSeries 400® (formerly called AS/400®)

 indicates information that is specific to WebSphere Commerce for AIX® .


 indicates information that is specific to WebSphere Commerce for Linux® .

 indicates information that is specific to WebSphere Commerce for Solaris Operating Environment software.



indicates information that is specific to WebSphere Commerce for Windows® .

 Professional indicates information specific to WebSphere Commerce Professional Edition.






 Business indicates information specific to WebSphere Commerce Business Edition.

Path variables

This guide uses the following variables to represent directory paths:

WC_installdir

This is the installation directory for WebSphere Commerce. The following are the default installation directories for WebSphere Commerce on various operating systems:

	AIX	/usr/WebSphere/CommerceServer55
	400	/QIBM/ProdData/CommerceServer55
	Linux	/opt/WebSphere/CommerceServer55
	Solaris	/opt/WebSphere/CommerceServer55
	Windows	C:\Program Files\WebSphere\CommerceServer55

 *WC_userdir*

This is the directory for all the data that is used by WebSphere Commerce which can be modified or needs to be configured by a user. An example of such data is WebSphere Commerce instance information. This directory is unique to OS/400®.

The *WC_userdir* variable represents the following directory:
/QIBM/UserData/CommerceServer55

WAS_installdir

This is the installation directory for WebSphere Application Server. The following are the default installation directories for WebSphere Application Server on various operating systems:

	AIX	/usr/WebSphere/AppServer
	400	/QIBM/ProdData/WebAS5
	Linux	/opt/WebSphere/AppServer
	Solaris	/opt/WebSphere/AppServer
	Windows	C:\Program Files\WebSphere\AppServer

WAS_userdir

This is the directory for all the data that is used by WebSphere Application Server which can be modified or needs to be configured by a user.

	400	/QIBM/UserData/WebAS5/Base/WAS_instance_name
---	-----	--

Chapter 1. Introduction to administering WebSphere Commerce

This book highlights the tasks commonly performed by the Site Administrator. The Site Administrator role discussed in this book is synonymous with the system administrator role found in many companies, and is referred to as the Site Administrator from this point onwards.

The Site Administrator is one of the defined default roles that comes with WebSphere Commerce. Typically, the Site Administrator installs, configures, and maintains WebSphere Commerce and the associated software and hardware. The Site Administrator responds to system warnings, alerts, and errors, and diagnoses and resolves system problems. This role typically controls access and authorization (creating and assigning members to the appropriate role), manages the Web site, monitors performance, and manages load balancing tasks. The Site Administrator may also be responsible for establishing and maintaining several server configurations for different stages of development such as testing, staging, and production. This role also handles critical system backups and resolves performance problems.

For more information on other WebSphere Commerce roles, see the WebSphere Commerce Production online help. The Site Administrator performs tasks in the following areas:

Access Management

- Users
- Organizations
- Roles
- Access groups
- Policies
- Resource groups
- Action groups

Security

- Account policy
- Password policy
- Account lockout policy
- Security checker

Performance

Monitor system performance

Configuration

- Transports
- Message types
- Message logging
- Tracing
- Component configuration
- Scheduler
- Registry
- About this product

WebSphere Commerce Payments

- Users
- Merchant settings
- WebSphere Commerce Payments Settings
- Cassettes
- Trace
- Access management
- Define the messaging systems for the site
- Propagate rules services to the production server
- Configure WebSphere Commerce Payments
- Configure and maintain the cache
- Configure the scheduler
- Configure logging
- Maintain the database
- Administer WebSphere Application Server
- Administer the Web server
- Monitor system performance
- Ensure the security of the site
- Troubleshoot

Administration tools

There are several tools that make the Site Administrator's tasks easier to perform. The main tools are described in this section. Subsequent chapters cover other tools and the tasks that can be done with them.

Administration Console

The Administration Console allows a Site Administrator to perform tasks related to site configuration and store configuration. For more details on using the Administration Console see, "Opening the Administration Console" on page 16. Tasks that the Site Administrator performs using the Administration Console, include the following:

- Manage users, organizations, roles, and member groups
- Manage access control
- Define transports and message types for the site
- Specify WebSphere Commerce Payments settings
- Enable and disable WebSphere Commerce components
- Schedule jobs to be run for the site
- Update registry components
- Publish a store archive

Store publish utility

The store publish utility is contained within the Administration Console and allows you to quickly create a store archive based on a sample provided with WebSphere Commerce. This utility also allows you to configure published stores by enabling or disabling certain features, (for example, collaboration) in your store. For more information on publishing a store, see "Publishing a store archive" in the WebSphere Commerce online help.

Configuration Manager

The Configuration Manager establishes and changes the infrastructure required to deploy and run WebSphere Commerce. Establishing the infrastructure involves acquiring information regarding the location of your database, Web server, and

WebSphere Commerce Payments, and any necessary information required to configure these applications for use with WebSphere Commerce. It also involves determining your machine's setup information, and any information necessary to initialize the WebSphere Commerce application itself. For more details on using the Configuration Manager, see "Launching WebSphere Commerce Configuration Manager" on page 12, or refer to the *WebSphere Commerce Installation Guide*.

The Configuration Manager allows the Site Administrator to perform administration tasks and configuration tasks without having to work with syntax-sensitive configuration files. These are some of the functions the Site Administrator can perform:

- Create or delete a WebSphere Commerce instance
- Change the configuration settings for a WebSphere Commerce instance
- Configure the Web server
- Configure the database to serve as a staging server
- Enter parameters for WebSphere Commerce Payments

WebSphere Commerce Payments

The WebSphere Commerce Payments is a component of WebSphere Commerce which provides secure, electronic payment processing to Internet merchants. Based on open standards-based technology, WebSphere Commerce Payments works with payment *cassettes* to support multiple payment protocols, including:

- VisaNet, a provider of worldwide telecommunications data and payment processing that has the ability to authorize and settle payments.
- BankServACH, a payment gateway that interfaces with the Automated Clearing House Network (ACH) to support online electronic check payments.
- Paymentech, a merchant processor of online credit card payments with direct links to Visa and MasterCard.
- Third-party payment cassettes written for WebSphere Commerce Payments.

Simply put, WebSphere Commerce Payments integrates with WebSphere Commerce and provides cash register-like functionality to manage payment processing. The customer never interacts with WebSphere Commerce Payments directly because WebSphere Commerce Payments sits behind the Internet merchant's storefront, receiving payments and processing those payments with banks and other financial institutions.

Note: IBM WebSphere Commerce Payments (hereafter called WebSphere Commerce Payments) was previously known as Payment Manager. Starting with version 3.1.3, the payments application was renamed to WebSphere Commerce Payments and references to the product were changed throughout this document.

Accommodating multiple payment methods

WebSphere Commerce Payments implements a multipayment framework architecture that provides a flexible and extensible way to accommodate merchants who need to accept multiple payment methods. The multipayment framework separates payment management, the *framework*, from specific payment methods, the software *cassettes*, so that each can evolve independently.

WebSphere Commerce Payments provides a plug-in architecture whereby software cassettes for each payment method attach to the payment framework. The

framework provides the generic infrastructure functions required for making and receiving payments using any payment method.

Payment cassettes are software applications that conform to the data flow and control conventions of the WebSphere Commerce Payments framework. Each payment cassette contains the implementation of specific payment methods and protocols.

Cassettes can be written by IBM or by third-party payment system implementors. IBM supports cassette development and offers detailed instruction to developers interested in writing their own payment cassettes. For more information on cassette development, see the *WebSphere Commerce Payments Cassette Kit Programming Guide* and downloads at:<http://www.software.ibm.com/commerce/payments/download.html>

WebSphere Commerce Payments user interface

You can access the Payments functions either through the WebSphere Commerce UI or through a standalone UI. Both can be used by Payments Administrators and individual merchants using hosted WebSphere Commerce Payments to:

- Configure WebSphere Commerce Payments
- Perform the following routine payment processing tasks in a payment-neutral way:
 - Approve orders
 - Deposit payments
 - Settle batches
 - Issue credits
 - View daily batch totals

The WebSphere Commerce Payments user interface is browser-based and can be accessed remotely and securely using the Secure Sockets Layer (SSL) capabilities of your Web browser.

WebSphere Commerce Payments roles

WebSphere Commerce Payments enforces roles such that each user is presented with a different view based on the user's role, for example, from the perspective of a Payments Administrator versus a Merchant Administrator. Within the merchant organization, WebSphere Commerce Payments enables the notion of different roles so that the merchant can monitor their own users. For example, a Clerk is restricted to operations such as approving an order, while a Merchant or Payments Administrator can modify a relationship with a financial institution.

There are four roles within the Payments framework which have relative mappings to corresponding roles in WebSphere Commerce. This is useful to know if you are an administrator who creates users and assigns roles. When you create users within the WebSphere Commerce Organization Administration Console, you must first assign those users the WebSphere Commerce roles listed below. Then the users will display in the Payments UI where you can assign them their corresponding Payments roles. The following table maps the Payments user roles to the corresponding WebSphere Commerce roles:

Table 1. Role mapping

Payments role	WebSphere Commerce role
Payments Administrator	Site Administrator

Table 1. Role mapping (continued)

Payments role	WebSphere Commerce role
Merchant Administrator	Site Administrator
Supervisor	Operations or Sales Manager
Clerk	Customer Service Supervisor

Both Payments Administrators and Merchant Administrators can manage WebSphere Commerce Payments. Supervisors and Clerks are financial roles. While they do not administer WebSphere Commerce Payments, they do manage the payment processing functions. The following table describes the responsibilities for each Payments role:

Table 2. Role responsibilities

Role	Responsibilities
Payments Administrator	<ul style="list-style-type: none"> • Define Merchant Administrators, Supervisors and Clerks • Configure merchants and their cassettes • Identify the Payments host name and status • Configure any installed cassettes • Add, delete and update event listeners • Settle payments • Approve or sale orders • Issue credits and reverse credits • Deposit orders • Search for orders and batches • View daily batch totals
Merchant Administrator	<ul style="list-style-type: none"> • Define Merchant Administrators, Supervisors and Clerks • Configure merchants and their cassettes • Add, delete and update event listeners
Supervisor	<ul style="list-style-type: none"> • Settle payments • Approve or sale orders • Issue credits and reverse credits • Deposit orders • Search for orders and batches • View daily batch totals
Clerk	<ul style="list-style-type: none"> • Settle payments • Approve or sale orders • Deposit orders • Search for orders and batches • View daily batch totals

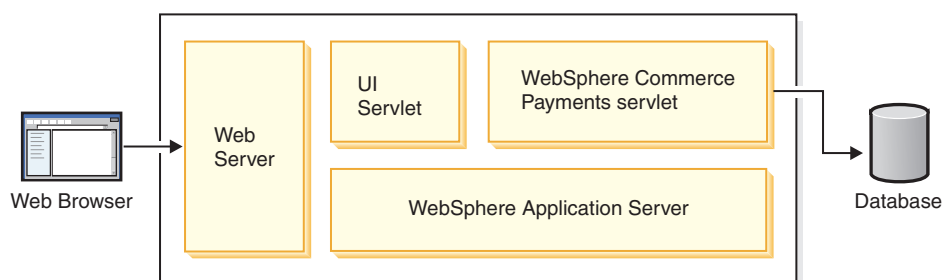
WebSphere Commerce Payments components

To understand WebSphere Commerce Payments it is useful to look at the following WebSphere Commerce Payments and define its components:

- Web server configured with WebSphere Application Server
- WebSphere Commerce Payments Servlets
- User Interface servlet

- Database

The following figure illustrates the internal components of WebSphere Commerce Payments:



The WebSphere Commerce Payments Servlet is the major component for WebSphere Commerce Payments. The Servlet is designed to work with WebSphere Application Server – a product that provides a common servlet environment for all platforms.

The remainder of this section further defines the WebSphere Commerce Payments Servlet as well as its ancillary support products:

- Web server
- WebSphere Application Server
- Database

Web server: Why a Web server? As mentioned earlier in this chapter, WebSphere Commerce Payments processes incoming HTTP requests. HTTP requests are first fielded by a Web server that is configured with WebSphere Application Server. These requests are then relayed to other WebSphere Commerce Payments components.

IBM WebSphere Application Server: Why configure your Web server with WebSphere Application Server? Two reasons:

1. WebSphere Application Server is a product that works with a variety of Web servers on multiple platforms.
2. WebSphere Application Server provides an environment where the WebSphere Commerce Payments servlets can be executed.

WebSphere Commerce Payments Servlet: The WebSphere Commerce Payments Servlet, also called the Payment Servlet, is the heart of WebSphere Commerce Payments. It receives requests from the user interface or other merchant applications and after performing security checks, invokes a Payment Cassette to handle any necessary payment processing such as communication with financial networks, and finally sends the responses back to the calling application. It is the only component that can connect to the WebSphere Commerce Payments database and access the secure data stored on it.

WebSphere Commerce Payments and e-commerce entities

In this section, you will see how WebSphere Commerce Payments supports the merchant's shopping software. Three scenarios demonstrate how WebSphere Commerce Payments, in conjunction with merchant software and generic payment protocols, interact with the following e-commerce entities:

- Merchant
- Buyer

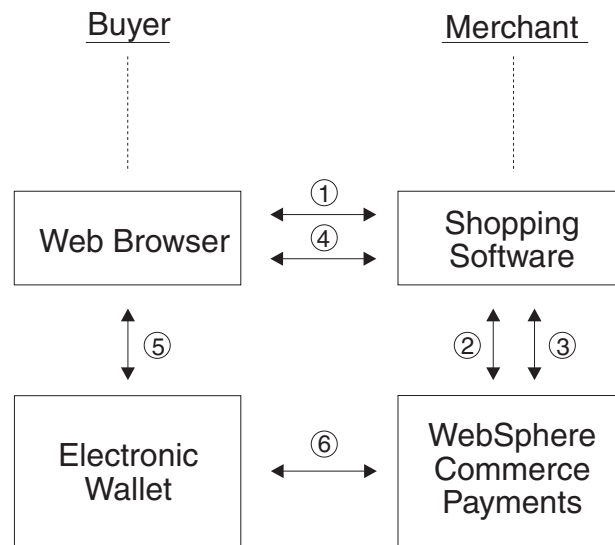
- Financial institution

The scenarios that follow illustrate e-commerce using WebSphere Commerce Payments. It is important to note that these scenarios are generic and do not represent any interactions specific to a particular payment protocol. Rather, they explore basic transaction flows common to many e-commerce business scenarios.

Scenario 1: cash and an electronic wallet: This first scenario uses a fictitious cash protocol and involves two entities: a buyer and a merchant. Funds are to be transferred from the buyer to the merchant. In this transaction, the buyer uses a protocol-specific electronic wallet application. The transaction steps are illustrated in the scenario diagram below as follows:

1. Interaction between the buyer and the merchant shopping software concludes with the buyer issuing a purchase request to the merchant.
2. In response to the purchase request, the merchant software sends a payment initiation command, that is, receive payment to WebSphere Commerce Payments.
3. WebSphere Commerce Payments responds with a payment initiation message which contains protocol-specific information for the buyer's wallet program.
4. The merchant software forwards this payment initiation message to the buyer's browser.
5. The payment initiation message received by the buyer's browser initiates the buyer's wallet program.
6. The buyer completes the transaction through interactions with WebSphere Commerce Payments and the wallet software.

Scenario 1: WebSphere Commerce Payments interacting with cash and an electronic wallet:



What separates this scenario from the two that follow, is that it demonstrates an e-commerce transaction involving only two entities:

- The buyer, represented by the electronic wallet
- The merchant using WebSphere Commerce Payments

No financial institution is involved.

Scenario 2: credit card and an electronic wallet: The second scenario illustrates a purchase using a fictitious credit card protocol and involves three entities:

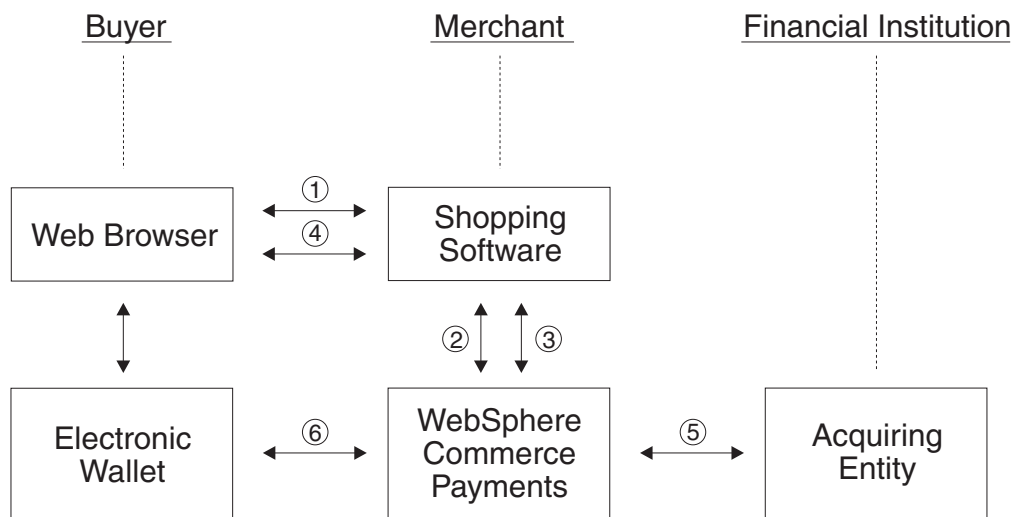
- Merchant
- Buyer, again using an electronic wallet
- Financial institution

This scenario flows similarly to scenario 1 with the following differences:

- When the buyer's wallet completes the purchase request, WebSphere Commerce Payments issues an approval request to the financial institution's acquiring entity to ensure sufficient cardholder funds.
- The transaction concludes with WebSphere Commerce Payments sending a confirmation response to the buyer's wallet indicating approval status.

The scenario diagram illustrates this variance in steps five and six below:

Scenario 2: WebSphere Commerce Payments interacting with a credit card and an electronic wallet:

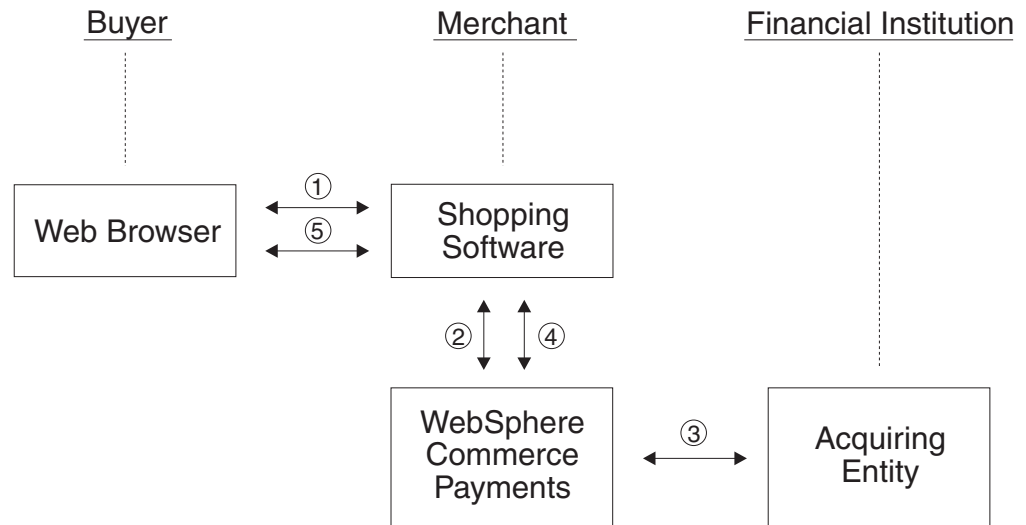


With the customer's wallet program initiated and the transaction completed, the financial institution's acquiring entity confirms sufficient cardholder funds, while WebSphere Commerce Payments confirms to the buyer that fund deposit has occurred.

Scenario 3: credit card and SSL: The final scenario illustrates a credit card protocol where the buyer has no wallet, or consumer-specific software. All consumer information is passed to the merchant software in a secure manner – such as in an SSL socket. The transaction steps are illustrated in the scenario diagram below as follows:

1. Interaction between the buyer and the merchant shopping software concludes with the buyer issuing a purchase request to the merchant.
2. In response to the purchase request, the merchant software sends a payment initiation command to WebSphere Commerce Payments.
3. WebSphere Commerce Payments attempts to approve the transaction, by confirming sufficient cardholder funds through the acquiring entity's financial institution.
4. WebSphere Commerce Payments returns a result, such as fund deposit has occurred, to the merchant software.
5. Finally, the buyer is notified of the purchase request's success or failure.

Scenario 3: WebSphere Commerce Payments interacting with a credit card and SSL:



The preceding diagrams illustrate possible e-commerce scenarios and demonstrate WebSphere Commerce Payments' role within those scenarios. Many other business flows are possible, and WebSphere Commerce Payments will assume a similar payment function in those interactions.

Organization Administration Console

The Organization Administration Console allows the Site Administrator to control the organizations that access your site or store. The tasks that you are authorized to perform display on the Organization Administration Console home page through various menus. These tasks are based on the user roles and authority levels. For more information, see the Organization Administration Console information in the WebSphere Commerce online help.

WebSphere Commerce Accelerator

The WebSphere Commerce Accelerator allows you to maintain your online stores by completing various store operations, from managing the look and feel of your store to creating and maintaining orders to tracking store activities. Tasks that you are authorized to perform in your role are displayed on the WebSphere Commerce Accelerator home page menus. These tasks are based on user roles and authority levels, which are defined by the Site Administrator by using the Administration Console. For more information refer to the WebSphere Commerce Accelerator information in the WebSphere Commerce online help.

Note: Tasks performed by the Site Administrator using the Organization Administration Console or the WebSphere Commerce Accelerator, are described in the online help.

WebSphere Application Server

The WebSphere Application Server is a Java-based application environment for building, deploying, and managing Internet and intranet Web applications. WebSphere Application Server also provides the run-time support for Java™ Server Page files.

The Administrator's Console in WebSphere Application Server provides the tools and features to allow an administrator to do the following:

- Stop and start the WebSphere Commerce Server
- Stop and start the IBM WebSphere Commerce Payments
- Install and configure resources (for example, servlets and enterprise beans)
- Assign security to resources
- Ensure that all applications are available
- Grant or revoke user access (for example, create a new account for a new employee and add the person to particular user groups)
- Monitor Commerce Server performance
- Clone application components for improved performance
- Provide traces and debugging information for running WebSphere applications

WebSphere Application Server information is available at the WebSphere Application Server Web site: <http://www.ibm.com/software/webservers/appserv>. For more detailed information on system administration for the WebSphere Application Server, refer to the following documents:

- *Systems Administration*
- *Tuning and Troubleshooting*

Database

You can use DB2[®] or Oracle as your database management system. For more information, refer to the *WebSphere Commerce Installation Guide* for each system.

Chapter 2. Basic administration tasks

This chapter covers the basic administration tasks that you are required to perform on a regular or one-time-only basis. Depending on your business process, you may not be required to perform each task.

One-time-only administration tasks

The following tasks are usually performed one time only.

Installation and configuration

One of your tasks as a Site Administrator is to install and configure the WebSphere Commerce site. Generally, this is the first step you will be required to perform. Installing usually involves installing the WebSphere Commerce software according to the operation system you are using. Installation steps can be found in the install guides that come with the product. For more information on installation and configuration, see the *WebSphere Commerce Installation Guide*. Further configuration information can be found in the WebSphere Commerce online help.

Regular administration tasks

The following tasks are performed on a regular basis:

- Starting and stopping WebSphere Commerce
- Opening Configuration Manager
- Creating, configuring, and deleting a WebSphere Commerce instance
- Opening the Administration Console
- Opening the Organization Administration Console
- Enabling and disabling WebSphere Commerce components
- Starting and stopping WebSphere Commerce Payments

Starting and stopping WebSphere Commerce

Each instance of WebSphere Commerce has its own WebSphere Commerce Server. The WebSphere Commerce Server is a Java-based commerce server used to control the flow of information in the WebSphere Commerce system. For more information on each of the components in WebSphere Commerce Server, see "WebSphere Commerce Server", in the WebSphere Commerce Production and Development online help.


A WebSphere Commerce instance is started and stopped as an application server called *WC_instance_name*, where *instance_name* is the name of the WebSphere Commerce instance you want to start or stop. For example, to start a WebSphere Commerce instance called demo, you would need to start an application server called WC_demo.

To start or stop a WebSphere Commerce instance, do the following:

1. Decide which WebSphere Commerce instance you want to start or stop.
2. Follow the instructions for stopping and starting an application server for your platform. For more information on these instructions, see "Starting and

stopping an application server on..<your platform>" in the WebSphere Commerce Production and Development online help.

If WebSphere Commerce Payments is not already started, you may want to start it after starting WebSphere Commerce. For instructions on starting WebSphere Commerce Payments, see "Starting and stopping WebSphere Commerce Payments" on page 18.

Note:  For instructions on starting WebSphere Commerce within WebSphere Commerce Studio, see "Starting and stopping WebSphere Commerce within WebSphere Commerce Studio" in WebSphere Commerce Production online help .

Launching WebSphere Commerce Configuration Manager

Configuration Manager lets you enable and configure various components of WebSphere Commerce. Whenever possible, components should be enabled through the Configuration Manager rather than by modifying the WebSphere Commerce configuration file. To create or configure a WebSphere Commerce instance, you should use the Configuration Manager.

Launching WebSphere Commerce Configuration Manager on AIX, Linux, and Solaris software

To launch WebSphere Commerce Configuration Manager on AIX, Linux, and Solaris software, do the following:

1. Start the server by doing the following:
 - a. Open a terminal window.
 - b. Log in as the WebSphere Application Server user created when the postinstall script was run as part of installing WebSphere Commerce. The default WebSphere Application Server user name is *wasuser*.
 - c. Issue the following commands:

```
export DISPLAY=host_name:0.0
xhost +host_name
cd /usr/WebSphere/CommerceServer55/bin
./config_server.sh
```



```
export DISPLAY=host_name:0.0
xhost +host_name
cd /opt/WebSphere/CommerceServer55/bin
./config_server.sh
```

where *host_name* is the fully qualified host name of the machine from which you want to access the Configuration Manager.

Notes:

- 1) Do not close the terminal window you entered the `config_server.sh` command in or the Configuration Manager server will stop.
 - 2) Do not run the Configuration Manager server as a background process – this is a potential security risk.
2. Start the client by doing the following:
 - a. Open another terminal window.

- b. Log in as the WebSphere Application Server user created when the postinstall script was run as part of installing WebSphere Commerce. The default WebSphere Application Server user name is *wasuser*.
- c. Issue the following commands:

▶ AIX ▶ Solaris

```
export DISPLAY=host_name:0.0
xhost +host_name
cd /usr/WebSphere/CommerceServer55/bin
./config_client.sh &
```

▶ Linux

```
export DISPLAY=host_name:0.0
xhost +host_name
cd /opt/WebSphere/CommerceServer55/bin
./config_client.sh &
```

where *host_name* is the fully qualified host name of the machine from which you want to access the Configuration Manager.

Launching WebSphere Commerce Configuration Manager on iSeries

To launch WebSphere Commerce Configuration Manager on iSeries, do the following:

1. Start the server by doing the following:
 - a. Log on to the iSeries ensuring that the profile has a *SECOFR user class, and is set up with with the language specific settings of either English, or the language that you will choose as the default language for your instance.
 - b. Start a QShell session by entering the following command:

```
STRQSH
```

and do the following in the QShell session:

- 1) Switch to the WebSphere Commerce server bin directory by issuing the following command:
- 2) Start the configuration manager server program by issuing the following command:

```
config_server.sh -port server_port_number
```

where *server_port_number* is the port number on the iSeries server to which Configuration Manager will listen. This parameter is optional, with the default being 1099. This value must be between 1024 and 65535, and not currently in use.

Note: If you are using a system where your primary language is not the same as the language in which you are creating your instance, then you must add the QSYSlanguage_feature_number library into your user profile's library list. Otherwise the profile will try to locate it under QSYS. To add the language feature library, use the EDTLIBL command.

- c. If this is the first time that Configuration Manager is run on the system, you will refer to the following messages:

Attaching Java program to /QIBM/ProdData/WebCommerce55/lib/wcsconfig.JAR.
Attaching Java program /QIBM/ProdData/WebCommerce55/lib/wcsruntime.JAR.
Attaching Java program to /QIBM/ProdData/WebCommerce55/wc.ear/lib
/wcslogging.JAR.
Attaching Java program to /QIBM/ProdData/WebCommerce55/lib/xml4j.JAR.
Attaching Java program to /QIBM/ProdData/WebCommerce55/lib/sslite.ZIP.

- d. When you receive the following messages, proceed to the next step:

Registry created.
CMServer bound in registry.

2. Start the client by doing the following:
 - a. Using a command prompt on the client machine, change to the WCS400 directory.
 - b. Configure the client by running the following command:
`config_client.bat iSeries_Host_name Server_port_number`

where

iSeries_Host_name
is the fully qualified host name of the server.

Server_port_number
is the port number on the iSeries server on which the Configuration Manager is listening.

Launching WebSphere Commerce Configuration Manager on Windows

To launch WebSphere Commerce Configuration Manager on Windows, do the following:

1. Ensure that the IBM WC Configuration Manager server process is running by selecting **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Services** and check that the IBM WC Configuration Manager service has a status of Started.

Important

Leaving the IBM WC Configuration Manager server service running could potentially pose a security problem. Stop the WC Configuration Manager server service when you are not using the Configuration Manager.

To prevent potential security problems, you should also ensure that the IBM WC Configuration Manager server is set for manual startup, not automatic.

2. Select **Start** → **IBM WebSphere Commerce** → **Configuration**.

Creating, configuring, and deleting an instance

To create, configure, or delete a WebSphere Commerce instance, you should use the Configuration Manager. This tool is designed to allow you to configure your instance while preventing you from entering invalid values.

The Configuration Manager should be used to modify most WebSphere Commerce configuration parameters. However, some parameters must be changed through the WebSphere Commerce Configuration File (*instance_name.xml*). Refer to an individual component's documentation in the online help to determine if it should be configured through the Configuration Manager or through the WebSphere Commerce configuration file (*instance_name.xml*).

Creating and configuring an instance

To create an instance, do the following:


1. Open the Configuration Manager. For instructions on opening Configuration Manager see, “Launching WebSphere Commerce Configuration Manager” on page 12.
2. Expand the node with your machine name, select **Instance List**, and from the Action menu select **Create instance**.
3. The instance creation wizard will guide you through the instance creation process. If you require assistance on any panel, select **Help**.


To configure an instance, do the following:

1. Open the Configuration Manager.
2. Expand the node with your machine name, expand **Instance List**, and select the instance you want to modify.
3. Select the component that you want to modify
4. If this component already exists, you can select it and the appropriate panel will open. You can then modify the component as needed.
5. If the component does not exist, you can right-click on the component, and select to create the component. A wizard will guide you through the component creation process.

Deleting an instance

To delete a WebSphere Commerce instance, do the following:

1. Back up your WebSphere Commerce instance (recommended).
2. Switch to the `WC_installdir\instances\instance_name\xml` directory,  `WC_userdir/instances/instance_name/xml`, and copy the `instance_name.xml` file to a backup directory.
3. Start the WebSphere Application Server.
4. Remove the WebSphere Commerce server by doing the following:


-  From QSHHELL session command line, type:
`WC_installdir/bin/rmCommerceServer.sh instance_name`

   From a command prompt, type:

```
WC_userdir/bin
./rmCommerceServer.sh instance_name
```

 From a command prompt, type:

```
WC_installdir/bin
./rmCommerceServer instance_name
```

5. In the WebSphere Commerce Configuration Manager, right-click your instance and select **Delete**.
6. Drop your WebSphere Commerce Business Edition database:
 -  If you are using a DB2 database, do the following:
 - a. Open a DB2 Command Window and type the following:

```
db2 drop db db_name
db2 uncatalog node node_name
```

where `db_name` is the name of the WebSphere Commerce database that you entered in the Configuration Manager, and `node_name` is the name of

the node you cataloged for for these databases. The default node name is the short version (that is, not fully qualified) of your machine's host name.

- ▶ **Oracle** If you are using an Oracle database, do the following:

- Start an SQL*Plus session and type the following:

```
sqlplus system/manager@wc_SID
```

where *wc_SID* is the Oracle System Identifier (SID) for your WebSphere Commerce database.

- Issue the following commands in the SQL*Plus session:

```
DROP TABLESPACE wc_tablespace_name
INCLUDING CONTENTS;
DROP USER wc_oracle_user_ID
CASCADE;
EXIT;
```

Note: You must also remove the actual tablespace files. The tablespace file is called *wc_SID.ora*, where *wc_SID* is the Oracle System Identifier (SID) for your WebSphere Commerce database.

- ▶ **400** Delete the instance database library by running the following SQL statement: drop schema *instance_name*. If you try to delete a schema and you get a message indicating that you cannot delete a receiver while it is attached, try the following command: ENDJRNPF FILE(*ALL) JRN(*instance_name*/QSQJRN).

- To completely delete your WebSphere Commerce instance, delete the following directory:

- ▶ **AIX** ▶ **Solaris** ▶ **Linux** ▶ **Windows**

```
WC_installdir/instances/instance_name
```

- ▶ **400** *WC_userdir/instances/instance_name*

- ▶ **400** Delete the instance user profile, using the DLTUSRPRF command.

Opening the Administration Console

Whenever you wish to perform an administrative task, you will likely be required to do it in the Administration Console. The Administration Console allows you to control your site or store by completing administrative operations and configuration tasks. For more information on the Administration Console, see "Administration Console" on page 2.

- Start the Administration Console by doing one of the following:

- Access the following URL in your browser:

```
https://host_name:8002/adminconsole
```

Where *host_name* is the fully qualified TCP/IP name of your WebSphere Commerce Server.

- ▶ **Studio** If you have WebSphere Commerce Studio installed and want to open the Administration Console from this machine, do the following:

- Ensure you have started WebSphere Commerce within WebSphere Commerce Studio. For instructions on starting WebSphere Commerce within WebSphere Commerce Studio, see "Starting and stopping WebSphere Commerce within WebSphere Commerce Studio" in WebSphere Commerce Production online help.

- Access the following Web address in your browser:

```
https://host_name/webapp/wcs/tools/servlet/ToolsLogon?
XMLFile=adminconsole.AdminConsoleLogon
```

where *host_name* is the fully qualified HTTP host name of your WebSphere Commerce Studio machine

2. Log on to the Administration Console by doing the following:
 - a. From the Logon page, type your Administration Console logon user name and password and click **Log On**. If you are authorized to work with more than one store or language, the Administration Console Site/Store Selection page displays. If you are authorized to work with a single store and language, the Administration Console home page displays. Tasks that you are authorized to perform display in the Administration Console home page.
3. From the Administration Console Site/Store selection page, select the **site** or **store** radio buttons and click **OK**.
 - If you selected **site**, the Site Administration Console home page displays.
 - If you selected **store**, you can choose the store name and language from the **Name** and **Language** drop-down boxes. The Administration Console home page displays.

Opening the Organization Administration Console

Log on to the Organization Administration Console by doing one of the following:

- Start the Organization Administration Console by doing one of the following:
 - Access the following URL in your browser:
`https://host_name:8004/orgadminconsole`

where *host_name* is the fully qualified TCP/IP name of your WebSphere Commerce Server.

- If you have WebSphere Commerce Studio installed and want to open the Organization Administration Console from this machine, do the following:
 1. Ensure you have started WebSphere Commerce within WebSphere Commerce Studio. For more information, see the WebSphere Commerce online help.
 2. Access the following Web address in your browser:
`https://host_name/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=buyerconsole.BuyAdminConsoleLogon`

where *host_name* is the fully qualified HTTP host name of your WebSphere Commerce Studio machine.

- To log on to the Organization Administration Console, do the following:
 1. From the Logon page, type your Organization Administration Console logon user name and password and click **Log On**. If you are authorized to work with more than one store or language, the Organization Administration Console home page displays. Tasks that you are authorized to perform display in the Organization Administration Console home page.
- To log on and change your Organization Administration Console password, do the following:
 1. From the Logon page, type your Organization Administration Console logon user name and password.
 2. Select the **Change password** check box and click **Log On**. The Change Password page displays.

3. In the **Old password** field, type your current Organization Administration Console logon password. This field accepts up to 128 alphanumeric characters.
4. In the **New password** field, type a new logon password. This field accepts up to 128 alphanumeric characters.
5. In the **New password confirmation** field, re-type the new logon password.
6. Click **Change** to save the new password. The Organization Administration Console home page displays. Tasks that you are authorized to perform display in the Organization Administration Console home page.

Enabling and disabling WebSphere Commerce components

As an administrator you may want to have certain components of WebSphere Commerce only available at certain times. As a result of differing customer needs, WebSphere Commerce allows you to enable or disable WebSphere Commerce components when needed, by using the **Configuration** menu in the administration console.

To enable or disable WebSphere Commerce components, do the following:

1. Open the Administration console and log on as a site administrator.
2. From the **Configuration** menu, click **Component Configuration**. A page listing all of the available components for your site displays.
3. To enable components do the following:
 - a. From the list of **Available components**, select those that you want to enable and click **Add**
 - b. Verify that all of the components you want enabled are in the **Selected components** list and click **OK**.

Note: Before enabling any component, refer to the documentation on that component to ensure that it is properly configured.

4. To disable components do the following:
 - a. Select the components that you want to disable from the list of **Selected components** and click **Remove**.
 - b. Verify that all of the components you want disabled are in the **Available components** list and click **OK**.

Starting and stopping WebSphere Commerce Payments

WebSphere Commerce Payments is a protocol-independent payment transaction server for online merchants. It provides cash register-like functionality to a site, supporting multiple payment methods using protocol-specific cassettes. These cassettes are software components that can be attached to the Payments framework to interpret generic payment and administrative commands into payment protocol-specific requests, which are then forwarded to the appropriate recipient, such as the payment gateway of an Acquirer institution. The end result is similar to when a cashier swipes a payment card at the checkout counter in a traditional store. Payments handles all the background details of Internet payments on behalf of the merchant (or a group of merchants, such as a store group), and provides a graphical interface to simplify the transaction management. Payments performs the following functions:

- It verifies, upon startup and dependent on the cassettes, that all required Acquirer certificates are present. If a certificate is missing, it is requested from the Acquirer or Payment Gateway automatically.

- It runs permanently, listening on dedicated ports and serving all incoming requests from the wallet and the merchant server.
- It communicates with the Acquirer over the Internet using the TCP/IP protocol or other protocols depending on the cassette.
- It uses database tables to maintain information on the status of transactions, approval requests, and deposit requests. The records in these tables are kept for working purposes and for tracking and record-keeping. The database also contains configuration tables that retain information about the merchant, the brands of payment cards, and the Acquirers that provide card authorization and payment capture services to the merchant.

The merchant server interacts with Payments by using Payments APIs. These APIs are designed to be general enough to support Secure Electronic Transactions (SET™) as well as other payment technologies.

As an administrator, you will be required to start and stop the Payments service. Starting and stopping the payment engine requires that you have a .payment file in the directory where Payments is installed. The .payment file is created during the installation of Payments.

Starting WebSphere Commerce Payments

To start WebSphere Commerce Payments do the following:

1. Start the WebSphere Application Server Administrative Console and make sure the WebSphere Payments Application server is started.

2. Log on as follows:



    Log in as your database instance ID.

 Log in as a user with Administrator's group privileges.

3. Go to the directory where WebSphere Commerce Payments is installed. The default install locations are as follows:

 /usr/lpp/WebSphere/CommerceServer55/payments

 /QIBM/ProdData/CommercePayments/V55

  /opt/WebSphere/CommerceServer55/payments

 drive:\Program Files\IBM\WebSphere Commerce Payments

4. Enter the following command:

   ./IBMPayServer

 IBMPayServer

 You must specify a password after running this command.

Note: If you have selected **Password Required** when configuring WebSphere Commerce Payments, you must invoke the IBMPayServer command in order to fully initialize the WebSphere Commerce Payments application. The IBMPayServer command is used to specify the WebSphere Commerce Payments password used to decrypt any sensitive data that is stored in the database. For more information on this command see, "Issuing the IBMPayServer command", in the WebSphere Commerce Production and Development online help. For more information on configuring WebSphere Commerce Payments, see "Configuring WebSphere Commerce Payments" on page 71. For detailed information on configuring WebSphere Commerce Payments, see Appendix C, "WebSphere Commerce Payments Tutorial," on page 155.

Stopping WebSphere Commerce Payments

To stop WebSphere Commerce Payments you need to do the following:

- Stop WebSphere Commerce Payments.
- Stop the Payment Manager application server under WebSphere Application Server.

To stop WebSphere Commerce Payments complete steps 1–3 for “Starting WebSphere Commerce Payments” on page 19, and use the following commands for step 4:

```
▶ AIX ▶ Solaris ▶ Linux ./StopIBMPayServer  
▶ Windows StopIBMPayServer
```

Note: For information on stopping Commerce Payments through Configuration Manager, or WebSphere Commerce Studio, see “Starting and stopping WebSphere Commerce Payments in WebSphere Commerce Studio, and “Starting and stopping WebSphere Commerce Payments from Configuration Manager”, in the WebSphere Commerce Production online help.

Administrative tasks in the install guides

The following administrative tasks can be found in the *WebSphere Commerce Install Guides*:

- Modifying a WebSphere Commerce instance or Commerce Payments instance
- Deleting a WebSphere Commerce instance
- Other command line configuration tasks
- Cataloging a remote DB2 database
- Generating WebSphere Commerce encrypted passwords
- Generating WebSphere Commerce Payments encrypted passwords

Chapter 3. Configuration

This section describes configuration tools and tasks that the Site Administrator uses to ensure optimum functioning of the WebSphere Commerce site.

Configuration services

Architectural configuration

The Site Administrator is responsible to work with an architect to implement a configuration in which WebSphere Commerce will operate. For example, WebSphere Commerce and the required stack (Web server, WAS, database, and Commerce Payments) may all reside on the same physical machine or node, leading to a single node architecture, or the various tiers (software products) of WebSphere Commerce may reside on different physical nodes (machines,) resulting in a multi-node architecture. Depending on the complexity of the solution there may actually be a clustered environment, with load sharing implemented between different machines or clones. The Site Administrator may also be responsible for establishing several server configurations for different stages of development, such as testing, staging, and production. For more information on the different configurations available with WebSphere Commerce, refer to the *WebSphere Commerce Installation Guide*.

Messaging services

The Messaging services interface in the Administration Console allows the Site Administrator to set up and manage the delivery of messages for the site and stores. The WebSphere Commerce messaging system allows you to manage all aspects of defining and sending messages that are generated within WebSphere Commerce. The messaging system can send messages by using transports such as e-mail and plain files. For e-mail the supported outbound protocol is SMTP; the message encoding depends on the specified language. Plain file messages use the UTF-8 encoding standard. Optionally, you can configure the messaging system to send messages to a back-end system by using WebSphere MQ and IBM CrossWorlds[®]. Messaging services can also be configured to send an outbound message and wait a period of time for the reply message. To configure the outbound messaging system use the Configuration menu in the Administration Console.

WebSphere Commerce uses a plug-in model that implements the Java 2 Enterprise Edition Connector Architecture (J2EE/CA) to provide a common interface between the system and the various transports. The use of a common interface with external transports allows the implementation details of the transport to be kept separate from the operation of the messaging system. This architecture makes it possible to plug in additional transports that adhere to the J2EE/CA interface. This way, you can easily customize the solution to fit your business needs and your environment.

Scheduling services

The scheduler is a component of a Commercet Server primarily used to schedule jobs and launch jobs that are based on a timing scheme. Each scheduled job runs as a separate thread. You can schedule multiple jobs to run simultaneously however, it is recommended to scatter jobs to control the workload and avoid long

periods of idle time. A job is a WebSphere Commerce command scheduled to run at a specified time or interval. To specify the timing for the job use the command start and interval parameters of the AddJob command. For more information on the AddJob command, see "AddJob command" in the WebSphere Commerce Production and Development online help.

The scheduler must be running in the following cases:

- IBM WebSphere Commerce Payments is running.
- An auction is running.
- A rules service is being propagated to the production environment
- The Site Administrator is changing the system and wants to communicate the changes to all clones, without restarting each instance of the cluster. .
- The Available To Promise (ATP) features are used.
- The site sends transacted messages.

If none of these situations apply, the Site Administrator may choose to disable the scheduler.


Logging services

The purpose of logging messages in the WebSphere Commerce server is to notify the administrator if unexpected errors or abnormal conditions occur in the WebSphere Commerce application. Message logs and traces are important diagnostic tools that aid the Site Administrator in determining the source of problems. Tracing is a problem-determination mechanism. Tracing assists developers in debugging the code during the development stage and assists the technical support team in solving customer problems. Since logging from WebSphere Commerce makes use of the logging facility from WebSphere Application Server, the Log Analyzer can be used. The Log Analyzer is a graphical utility that facilitates viewing and analyzing log files. For more information on the Log Analyzer, see the WebSphere Application Server InfoCenter .

WebSphere Commerce provides facilities for logging. For existing customers, we continue to support ECTrace and ECMessage. For new implementations, we recommend using JRAS which is provided by WebSphere Application Server, and useable by WebSphere Commerce applications.

The WebSphere Commerce Log API's are:

- ECTrace traces the data flow. The trace entries are captured in a file for debugging purposes.
- ECMessage logs diagnostic messages. Messages are locale-sensitive, and are stored in log files in the file system. If error notification is enabled, technical support receives alert notifications. Diagnostic logs are used for problem determination. By default, the log file name is `activity.log`.
- JRas a standalone logging toolkit that provides message logging and diagnostic trace primitives.

The location of the default output files is `WAS_installdir/logs/WC_<instance name>/.`  `WAS_userdir/logs/WC_instance_name/.` The default output files are:

- `native_stderr.log` is a process log that contains text written to the stderr stream.
- `native_stdout.log` is a process log that contains text written to the stdout stream.

- startServer.log is the log when starting the server.
- stopServer.log is the log when stopping the server.
- SystemErr.log logs any system error while the server is running.
- SystemOut.log logs the system output file while the server is running.
- activity.log logs continuous activity. This log is located in the *WAS_installdir/logs* directory.

For information on the WebSphere Application Server log API's, see the section on logging and tracing in the WebSphere Application Server InfoCenter.

ECMessage

Diagnostic logs are used for problem determination. ECMessageLog logs diagnostic messages, and ECMessages are localized. ECMessages are split into the following categories:

- System messages
- User messages

System and user messages: System messages appear in the logs and are for the benefit of debugging problems. System messages provide diagnostic information for Site Administrators. These messages may follow a system malfunction, or indicate some other significant event.

System messages are assigned a product-specific message ID:

CMNnnnns

where

nnnn: the key number used to identify what is affected.

s: the severity of the message.

User messages are frequently shown on the browser, and are for the benefit of a customer visiting the site. User messages will give details about the problem for example, they will indicate whether a parameter specified is invalid, which indicates to the customer what value to fix when they resubmit the request. Site Administrators can use the message ID to look up more details associated with that message; customers can report the message to support personnel for troubleshooting.

Logging levels: There are five logging levels, or severities, in the WebSphere Commerce logging system. The system message severities are: error, status, warning, information, and debug.

- Error messages are logged at all times by default. Error messages expose an error condition that can lead to system malfunction. An error message can be sent as an e-mail, WebSphere MQ message, or by another form of notification to a Site Administrator registered with messaging.
- Status messages indicate certain states reached by WebSphere Commerce. For example, each component should persist a message when loaded or when the message has moved to a specific state. Status messages assist technical support in understanding the state of the components and application.
- Warning messages reveal a potential problem.
- Information messages follow events that occur within the WebSphere Commerce system. Information messages are related to events that trigger changes in the system state. For example, an information message is persisted when an order has been submitted.

- Debug messages overlap with the trace component. However, debug messages allow Site Administrators to investigate a problem themselves, without involving technical support.

Since WebSphere Commerce uses the WebSphere Application Server logging facility, and the WebSphere Application Server only has three types of logging levels, the WebSphere Commerce logging levels are mapped to the WebSphere Application Server as follows:

Table 3.

Logging levels in WebSphere Commerce	Logging levels in WebSphere Application Server
ERROR/ERR	TYPE_ERROR/TYPE_ERR
INFORMATION/INFO	TYPE_INFORMATION/TYPE_INFO
DEBUG	TYPE_INFORMATION/TYPE_INFO
WARNING/WARN	TYPE_WARNING/TYPE_WARN
STATUS	TYPE_INFORMATION/TYPE_INFO

The WebSphere Application GUI is used to control the enablement of the logging level or severity type. Site Administrators can specify which logging severities to record in the WebSphere Application Server Administration Console. For more information on enabling logging levels, see the information in the WebSphere Application Server InfoCenter .

ETrace

Tracing is used for problem determination. Tracing assists developers in debugging code during the development stage and assists the technical support team in solving customer problems.

Tracing data is persisted for future reference in a trace file. A data structure consists of context information, such as a class name, a method name, and a text message. Multiple data structures describe the data flow within a software application. By analyzing the data structure sequence, a developer can understand the path that was executed, which can help determine the cause of malfunctions.

JRas

JRas consists of multiple java packages that provide message logging and diagnostic trace primitives. These primitives are not tied to any particular product or platform. JRas is basically composed of several components:

- **Loggers:** A logger is the primary object with which the user code interacts. There are two types of loggers: message loggers, and trace loggers. Message loggers create only message records, and trace loggers create trace records. A logger contains one or more Handlers to which it forwards events for further processing.
- **Handlers:** A handler receives events from a logger, and provides an abstraction over an output device or event consumer. An example is a file handler, which knows how to write an event to a file.
- **Formatters:** Handlers are configured with Formatters, which know how to format events of certain types.
- **Event Types -** Messages and traces have their own pre-defined event types.
- **Event Classes -** The standalone JRas logging toolkit defines both message and trace event classes.

Note: For more information on the JRas logging toolkit, see the WebSphere Application Server JRas documentation in the WebSphere Application Server Information Center.

WebSphere JRas extensions: In order to integrate into the WebSphere Application Server run time or for usage in a J2EE environment, WebSphere provides a set of extension classes. Logging from WebSphere Commerce makes use of the WebSphere Application Server logging facility, and these extension classes provide a better correlation of messages and traces generated from different WebSphere products. This collection of extension classes are referred to as WebSphere JRas extensions. WebSphere JRas extensions provide appropriate logger implementation classes. Instances of these message and trace logger classes are obtained directly and exclusively from the WebSphere Manager class which is located in the `com.ibm.websphere.ras` package. Other components such as Payments and the JCA messaging framework, also make use of WebSphere JRas extensions. For a list of the WebSphere JRas extensions associated with WebSphere Commerce and Payments tracing components, see “Enabling tracing components” on page 61.

Note: WebSphere Commerce provides a wrapper for `ECMessage` and `ECTrace` that calls the WebSphere JRas extension API's, however it is recommended that Site Administrators call the JRas API's directly.

WebSphere Commerce Payments Problem determination

As you start to use WebSphere Commerce Payments, you may need to diagnose problems that you encounter while actually using the application. The WebSphere Log Analyzer provides an interface with which you can analyze the error and trace logs to determine a course of action. All of the Payments error and trace logs are available through the Log Analyzer except for the following messages:

- Third party cassettes
- Standalone Payments UI
- SampleCheckout
- Cashier

Rules services

The WebSphere Commerce uses rule services to interact with the Advisor Rule Server. A rule service acts as an interface to facilitate communication between the two applications. The rule service also provides a convenient way to update the rule-based portion of your site without having to stop the entire WebSphere Commerce. A rule service exists in a one-to-one relationship with a campaign. Each campaign published using the WebSphere Commerce Accelerator must have a corresponding rule service. This design provides flexibility as each marketing service, and therefore campaign, can be started, stopped, and refreshed independent of every other service. Whenever a customer profile, campaign, or rule service is updated using the WebSphere Commerce Accelerator, it is necessary to transfer the files to the appropriate location on the production server, and refresh the rule service. Each application clone, or Java Virtual Machine has a local instance of the rule server. While the need to communicate across application clone boundaries is minimal, there is one case where this is necessary: updating the status of a rule service. In order to receive a status report from each application clone, the system uses the Scheduler to broadcast a request to update the status of a rule service. The results are collected and displayed when the user presses the “View Status” button.

The Advisor Rule Server

WebSphere Commerce uses the Advisor Rule Server to process rules and provide personalized marketing content, consisting of advertisements and suggestive selling techniques. This server is incorporated into the WebSphere Commerce Server. The rule server is called by the WebSphere Commerce Server which passes information based upon the current shopping environment. The rule server processes this information against a collection of rules, created by the merchant or the marketer, and compiles appropriate output for the particular circumstances. The output is based upon criteria defined using the WebSphere Commerce Accelerator. The rule server is governed by certain limitations, contained in the License Information booklet. You require a separate license from HNC Inc, to exceed these limitations.

LDAP — Lightweight Directory Access Protocol

LDAP is a client-server protocol for accessing a directory service. It was initially used as a front-end to X.500, but can also be used with stand-alone and other kinds of directory servers. LDAP is used as a centralized information repository to support information sharing among various applications.

The LDAP information model is based on an entry, which contains information about some object (for example, a person). Entries are composed of attributes, which have a type and one or more values. Each attribute has a syntax that determines what kind of values are allowed in the attribute and how those values behave during directory operations. Examples of attribute syntax's include IA5 (ASCII) strings, JPEG photographs, u-law encoded sounds, URLs, and PGP keys. In general, an entry is uniquely identified by its distinguished name, or DN.

In WebSphere Commerce, a registered user has a unique identifier which is a string that has the same format as a distinguished name. If the WebSphere Commerce database is used as the member repository, the DN would be of the form 'uid=logonIDvalue' followed by the DN of the parent organizational entity that the user belongs to. So if a user has logonID 'john' and he belongs to the SWG organizational unit within the IBM organization, his unique identifier would be 'uid=john, ou=SWG, o=IBM, o=Root Organization'. If the directory server is used as the member repository, the unique identifier of the registered user would be his DN in the directory server, in which case the 'o=Root Organization' portion may not be part of his DN because WebSphere Commerce supports existing user entries in the directory server which may not be under the Root Organization.

WebSphere Commerce also supports storing organizational entities in the directory server and every organizational entity has a unique identifier which has the same format as a DN. If the WebSphere Commerce database is used as the member repository, the unique identifier would be of the form 'ou=organizationalEntityName' followed by the DN of the parent organizational entity. So if an organizational unit called Marketing is within the SWG organizational unit within the IBM organization, the unique identifier would be 'ou=Marketing,ou=SWG,o=IBM,o=Root Organization'. If the directory server is used as the member repository, the unique identifier of the organizational entity would be its DN in the directory server, in which case the 'o=Root Organization' portion may not be part of its DN because WebSphere Commerce supports existing organizational entries in the directory server which may not be under the Root Organization. Note also that whether 'o' or 'ou' is used within the DN depends on whether the organizational entity is an organization or an organizational unit.

When a directory server is used as the member repository, users and organizational entities can be stored in the directory server and you can configure which LDAP attribute should be used as the RDN attribute and which WebSphere Commerce attribute provides the value for the RDN attribute. A typical DN is as follows:

```
uid=jsmith, l=Toronto, st=Ontario, c=CA
```

where

uid This is the unique id for the user. The uid attribute in the above example is known as the RDN attribute. It uniquely identifies an LDAP entry under a parent entry with DN *l=Toronto, st=Ontario, c=CA*. In WebSphere Commerce, if the authentication challenge is logonID and password, WebSphere Commerce will do a search on the LDAP server under a given search space for a node with RDN = logonID. If X.509 certificate is the challenge type, WebSphere Commerce will use the subject name from the certificate as the RDN attribute value.

l The user's locality, or city.

st The user's state or province.

c The user's country or region.

The following IBM(R) Redbooks contain additional information about LDAP:

- *LDAP Implementation Cookbook*
- *Understanding LDAP*

Redbooks are available from the following site: <http://www.redbooks.ibm.com>.

To use SSL between the WebSphere Commerce Server and the LDAP server, refer to instructions provided by your LDAP server to perform the appropriate setup (for example, creating a key ring file) and specify any environment properties necessary for the WebSphere Commerce Server by using the JNDIEnvPropNameX and JNDIEnvPropValueX attributes in your *instancename.xml* file.

Authentication with LDAP

User profile information can be stored in either the WebSphere Commerce database or on the directory server. The authentication options are:

- LDAP
A user can logon with either his DN or RDN value and a password. If RDN is used, the user is searched using the search bases configured in the *ldapentry.xml* file. The user is authenticated against the LDAP server. User information on the LDAP server is replicated to the WebSphere Commerce database for run-time operational use.
- Database
The user is authenticated against the WebSphere Commerce database using the logon ID and password provided by the user.
- Other
Authenticate using a third party interface, store profile data in the WebSphere Commerce database

To specify the authentication mode use the Member subsystem page in the Configuration Manager. X.509 certificates can be used with either LDAP or database authentication, if X.509 certificates are used the Web server authenticates the user. In this case the Authentication mode setting determines where profile data is stored, LDAP or the WebSphere Commerce database.

LDAP registry

In WebSphere Commerce, you can use either LDAP or the WebSphere Commerce database as the member repository. Users and organizational entities can be stored in the LDAP server. Member groups currently can only be stored in the WebSphere Commerce database. If you use LDAP as the registry this corresponds to the following settings in the *instancename.xml* file: `AuthenticationMode=LDAP` and `ProfileDataStorage=LDAP`.

Data is replicated between the LDAP server and the WebSphere Commerce database. You can configure what data gets replicated using the *ldapentry.xml* file.

The following table describes the level of LDAP service offered in WebSphere Commerce:

Table 4. User Registration

	LDAP as registry	No LDAP support
New user registration or update of registered user's information through WebSphere Commerce	User information is created or updated in both the WebSphere Commerce database and the LDAP server.	User information is created or updated in the WebSphere Commerce database.
LDAP user registers or updates information from another LDAP application	User information is replicated to the WebSphere Commerce database only when the user logs on to the WebSphere Commerce Server, or when the user's information is needed by logic within the WebSphere Commerce Server.	N/A
Availability of LDAP server	The LDAP server must always be available.	N/A
LDAP connection fails	An error page is displayed to the user. An error message is logged in the system log file.	N/A

Table 5. Organization Entity Registration

	LDAP as registry	No LDAP support
New organizational entity registration or update of organizational entity information through WebSphere Commerce	Organizational entity information is created or updated in both the WebSphere Commerce database and the LDAP server.	Organizational entity information is created or updated in the WebSphere Commerce database.
Organizational entity information is created or updated from another LDAP application	Organizational entity is created or updated in the WebSphere Commerce database only when the organizational entity information is needed by logic within the WebSphere Commerce Server.	Not applicable.

Users whose registration are pending approval are also stored in the directory server. Context-dependent attributes are not stored in the directory server by default.

Only the user registration address is persisted to LDAP. The friends, relatives, or business associates address in the LDAP user's address book are not persisted.

Note: If you have chosen an LDAP server as the member repository, you should not use the Loader package to mass load users and organizational entities information into the WebSphere Commerce database. In WebSphere Commerce 5.5 we have introduced business models which will massload data into the database. For more information on the business models, see *WebSphere Commerce Store Development Guide*.

LDAP configuration parameters

After you install the LDAP system you want to use, you need to complete the appropriate fields in the Member Subsystem page of the Configuration Manager, or the *instancename.xml* file if you are configuring LDAP support manually. This file describes the parameters in the MemberSubSystem section of the *instancename.xml* file. Note that each instance of WebSphere Commerce contains a *instancename.xml* file. To avoid possible conflicts, ensure that the LDAP configuration information is the same for each instance.

Note: The following is example code; the values are fictitious and only for the purposes of this example. For your own file, you must provide valid values for the parameters.

```
<MemberSubSystem name="Member SubSystem"

    ProfileDataStorage="LDAP"

    AuthenticationMode="LDAP">

<Directory LdapPort="636"

    LdapType="SECUREWAY"

    LdapAuthenticationMode="SIMPLE"

    EntryFileName="d:/WebSphere/CommerceServer55/
xml/ldap/ldapentry.xml"

    LdapAdminPW="EaDPFd9VAf0="

    LdapVersion="3"

    LdapHost="basswood.torolab.ibm.com"

    SingleSignOn="0"

    LdapAdminDN="cn=root"

    display="false"

    MigrateUsersFromWCdb="ON"

    JNDIEnvPropName1="java.naming.referral"

    JNDIEnvPropValue1="ignore"

    JNDIEnvPropName2="java.naming.security.
protocol"

    JNDIEnvPropValue2="ssl"

    ...

    JNDIEnvPropNameN="java.naming.language"

    JNDIEnvPropValueN="en-US"

    LdapTimeOut="0" />

</MemberSubSystem>
```

Parameter descriptions

ProfileDataStorage

The value of this attribute determines where profile data for users and

organizational entities are stored. Note that even if the value is LDAP, some of the profile data is replicated to the WebSphere Commerce database.

AuthenticationMode

The value of this attribute determines the system on which users are authenticated. Valid values are:

- LDAP (default)
- DB
- Other (use for a third party mechanism)

LdapAdminDN

The DN of administrator used to perform operations on LDAP. This DN must be set up on LDAP and given permissions on LDAP.

LdapAuthenticationMode

The value of this attributes depends on which directory server you are using.

Simple and None are both valid for IBM SecureWay® and Netscape iPlanet.

- Simple (default)
- None

LdapTimeOut

The time in seconds before an LDAP search times out. A timeout value can also be specified on the LDAP server. The minimum of the two value takes precedence. If the value of LdapTimeOut is 0, then the search timeout is solely determined by LDAP server configuration. The default value is 0.

LdapVersion

The LDAP protocol version number. The only valid value is 3.

EntryFileName

The XML file that contains information on which WebSphere Commerce attribute maps to which LDAP attribute.

The default is:



```
WC_installdir/  
xml/ldap/ldapentry.xml
```



```
WC_installdir/xml/ldap/ldapentry.xml
```

LdapPort

The port used by LDAP server. The default value is 389.

SingleSignOn

A value of '0' means single sign on is turned off. '1' means it is turned on.

LdapAdminPW

No default - blank

The encrypted password of the Administrator. If the XML is modified manually, it is generated using `wcs_encrypt.bat` or `wcs_encrypt.sh`.

LdapHost

The host name of LDAP server. The default value is the current host name.

MigrateUsersFromWCdb

This option should be turned ON to ensure that the bootstrap users and

any organizations created by the store publish process get pushed to LDAP. Otherwise, this option should be turned OFF. Default value is OFF.

LdapType

The LDAP directory server that you are using, such as

    IBM SecureWay V5.1 (default).



- OS400 - DIRECTORY SERVICES
- IBM SecureWay V3.2.2

JNDIEnvPropNameX and JNDIEnvPropValueX

Use these attributes to specify any JNDI environment properties that you want the WebSphere Commerce server to set. These attributes must be specified manually in the `instancename.xml` file.

Note the following changes from WebSphere Commerce 5.1:

- MappingFileName is changed to EntryFileName
- ldapmap.xml is changed to ldapentry.xml
- LdapPersonRDN, LdapPersonSearchRoot, LdapPersonDefaultBase, LdapPersonOCS, LdapOrgOCS and LdapOrgUnitOCS are now part of the ldapentry.xml file and their names have been changed to the following:
 - LdapPersonRDN is now rdnName
 - LdapPersonSearchRoot is now searchBase
 - LdapPersonDefaultBase is now defaultBase
 - LdapPersonOCS is now objClass
 - LdapOrgOCS is now objClass
 - LdapOrgUnitOCS is now objClass
- Only LDAP protocol version 3 is supported

For information the `ldapentry.xml` and `ldapmap.dtd` files, see Appendix B, “LDAP files,” on page 149.

LDAP design features

The following information highlights some of the design features of the LDAP implementation in WebSphere Commerce.

- Multiple WebSphere Commerce attributes can be replicated to a single LDAP attribute with a separator character specified.
- You can replicate from Multiple LDAP attributes to a single WebSphere Commerce attribute. To do this use multiple `<map>` sections in the `ldapentry.xml` file; however, you should be aware that overwrites can occur.
- A single WebSphere Commerce attribute can be replicated to Multiple LDAP attributes.
- A single LDAP attribute can be replicated to multiple WebSphere Commerce attributes.
- Only the default address for the user is persisted to LDAP. The default registration address is defined as the address in the user’s address book with `selfaddress=1` and `logonid=nickname`. Where `logonID` is the `logonid` in the `USEREG` table and `nickname` is `NICKNAME` in the `ADDRESS` table. Other addresses in the user’s address book will not be stored on LDAP.
- Only registered users from WebSphere Commerce will be persisted to LDAP, not guest users.

- WebSphere Commerce allows multiple search bases to be specified when searching for a user entry on LDAP. However, if multiple user entries are found which satisfy the search criteria, an error will be generated.

The following table describes some of the limitations of LDAP support in WebSphere Commerce.

Limitation	Description
Multi-valued attributes in LDAP	WebSphere Commerce does not store or retrieve multi-valued LDAP attributes, however multi-valued attributes in LDAP are tolerated in that the first value is retrieved. During an update, if the new value already exists on LDAP no update is performed; otherwise, the first value of a multi-valued LDAP attribute is updated.
LDAP referrals and references	WebSphere Commerce does not support LDAP referrals and references.
Multi-component RDN	Only one LDAP attribute can be specified as the RDN attribute, not a combination of LDAP attributes.

Staging server

Most online stores operate 24 hours a day, 365 days of the year, making it difficult to perform maintenance or test changes to the system. The WebSphere Commerce staging server allows Site Administrators to copy their production database to a staging database in order to test updates without affecting customers. This is useful for testing updates to the product catalog, but it is also important for testing new shopping process commands.

The staging server consists of the following components:

A WebSphere Commerce instance

Tests and modifies your data.

Database schema scripts

Creates the staging tables and triggers for the staging database. The staging database contains the same schema and tables as the production database, plus a set of triggers to log changes made in the staging database. The staging database schema scripts add triggers to the database.

Changes are logged to the STAGLOG table (a staging table) using database triggers. Whenever you change a database table record in the staging database, the STAGLOG table records this change.

The Stage Copy utility

Allows an administrator to copy data from the production database to the staging database. You can copy the data into site-related tables, merchant-related tables, or individual tables.

The Stage Propagate utility

Allows an administrator to propagate changes from the staging database to the production database. The information in the STAGLOG table identifies the records in the staging database that must be inserted, updated, or deleted in the production database. The identified records are then

updated in the production database. Processed records are indicated in the STAGLOG table by a 1 in the STGPROCESSED column.

The Stage Check utility

Allows an administrator to check for potential unique index key conflicts between two tables on a staging server and a production server.

Staging server limitations

Before using the staging server, you should be aware of the following limitations:

- You cannot use the staging server with the buyer organization self-administration features.
- The member_id column of all staging tables (excluding MEMBER, MBRREL, MBRROLE, and MBRATTRVAL) must be organization or member groups, and not the user.
- For all site tables, the member_id must be -2001 or 0. For all tables containing both site and merchant data, the member_id for rows related to site data must be 0 or -2001.
- You cannot use the Stage Copy command if you are using RFQ features on your production system. You must use the push mode, for which the Stage Copy command is not available. Before you launch your production site, create the staging server and set up staging database. Deploy and test your data on the staging server, then push to the production server using the Stage Propagate command.
- You cannot create or update RFQs on a staging server.

Key splitting: In the reseller marketplace, IBM and the Site Administrator update data on the staging server, while resellers update data on the production server. This potentially causes a primary key collision.

WebSphere Commerce uses a key manager to generate primary keys for tables, and the key range is defined in the KEYS table. If the production server and staging server use the same key range, the key manager can allocate the same primary key value for the same table, on both the production and staging server, causing primary key collision. As a result, the key range must be split immediately after the staging instance and production instance are created. This ensures that the staging server and production server will use different key ranges.

Currently, WebSphere Commerce uses the following SQLs to split the key range on the staging and production servers. The SQL statements split the current key range into three equal portions.

Portion one

```
update keys set upperbound=(upperbound-lowerbound)/3 + lowerbound
where tablename in (select tablename from stgmertab) or tablename
in (select tablename from stgsitetab)
```

Portion two

```
db2 update keys set upperbound = (upperbound-lowerbound)/3*2 + lowerbound,
lowerbound = (upperbound-lowerbound)/3 +lowerbound+1, counter =
counter+(upperbound-lowerbound)/3 +1 where tablename in
(select tablename from stgmertab)
or tablename in (select tablename from stgsitetab)
```

Portion three

```
db2 update keys set lowerbound = (upperbound-lowerbound)/3*2 + lowerbound +1
, counter = counter +(upperbound-lowerbound)/3*2 +1 where tablename in
(select tablename from stgmertab) or tablename in (select tablename from stgsitetab)
```

The first SQL statement, portion one, must be run on the production server. This ensures that the production server will occupy one third of the full key range. The second SQL statement, portion two, must be run on the staging server. This means that the staging server will occupy the second one third of the full key range. The third SQL statement, portion three, can be kept for a second potential staging server in the future.

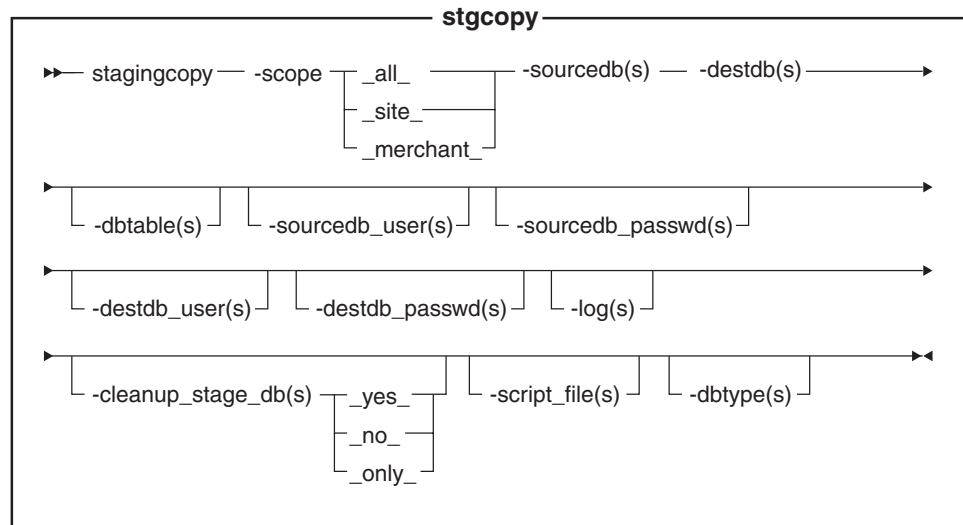
Note: The above key splitting SQLs are already integrated with the instance configuration process. You do not need to run them manually.

Staging server commands

Stage Copy utility command (Windows 2000, AIX, Linux, iSeries, and Solaris):

The Stage Copy utility copies data from the production database to the staging database. Note that you cannot use this command if RFQs are on your production system. For details, refer to “Staging server limitations” on page 34.

To run the Stage Copy utility, type the following from a command line on a machine that can connect to both the staging server and the production server database. Type the entire command on one line. It is shown here on more than one line for presentation purposes only.



Notes:

1. **Oracle** You must include the optional parameters, logon user ID and password in the command even if you are currently running this utility with the same user ID.
2. **400** iSeries now uses `stagingCopy.sh`. To run shell scripts on iSeries do the following:
 - a. Log on as a user profile that has a CCSID other than 65535.
 - b. Open a QSHLL command window by typing the following command on an OS/400 command line: `STRQSH`.
 - c. Run the command as follows:


```
/QIBM/ProdData/CommerceServer55/bin/stagingCopy.sh (parameters . . .)
```

Note: The user parameters must always be specified.

3. **z/OS** If you are working with DB2 for z/OS, you need to provide some additional parameters relative to the above syntax:

- `-dbtype DB2/390`
- `-sourcedb_user Source_DB_user`
- `-sourcedb_password Source_DB_password`
- `-sourcedb_schema Source_DB_schema` (for DB2 for z/OS databases only)
- `-destdb_user Destination_DB_user`
- `-destdb_password Destination_DB_password`
- `-destdb_schema Destination_DB_schema` (for DB2 for z/OS databases only)

Note the following:

- You need to specify the following parameters that are exclusively for Linux Hybrid:
 - `-sourcedb_schema`
 - `-destdb_schema`
- Both users (specified by `-sourcedb_user` and `-destdb_user`) need to have the DBADM privilege on the database objects created by each other
- The following parameters are required (and not optional as on other platforms):
 - `-sourcedb_user`
 - `-sourcedb_passwd`
 - `-destdb_user`
 - `-destdb_passwd`
- The DB type specified by `-dbtype` must be DB2/390

Parameter values

scope The level of scope for the copy to the staging server. Specify one of the following:

- `_all_`
Type `_all_` to copy both records related to the site and to all merchants.
- `_site_`
Type `_site_` to copy only site-related records.
- `_merchant_`
Type `_merchant_` to copy only records related to all merchants.

sourcedb

The name of the database on the production server.

400 The name of the database on the production server, as displayed in the relational database directory.

Oracle Use `host:port:sid`. For example, `myhost:1521:mydb`.

destdb The name of the database on the staging server.

400 The name of the database on the production server, as displayed in the relational database directory.

Oracle Use `host:port:sid`. For example, `myhost:1521:mydb`.

dbtable

(Optional) The name of any specific table to be copied. All records in this table will be copied, provided the records are within the scope specified by the scope parameter; otherwise, no records will be copied.

sourcedb_user

(Optional) The logon ID of the database administrator who has created the source database schema. If not specified, the ID of the user currently invoking the utility is used.


 (Required) The user profile associated with the commerce instance. This is the same as the source database schema.

sourcedb_passwd

(Optional) The password of the logon ID that is specified by the `sourcedb_user` parameter.

destdb_user

(Optional) The logon ID of the database administrator who has created the destination database schema.

 (Required) The user profile associated with the commerce instance. This is the same as the destination database schema.






Note: If not specified, the ID of the user invoking the utility is used.

destdb_passwd

(Optional) The password of the logon ID that is specified by the `destdb_user` parameter. If not specified, the system prompts you to enter the password.

log

(Optional) The path and name of the file in which the Stage Copy utility records its activities and errors. If this parameter is not specified, a log file called `stagingcopy_yyyy.mm.dd_hh.mm.ss.zzz.log` is created in the following log directory.

   `WC_installdir/logs`
 `WC_installdir\logs`
 `WC_userdir/instances.`

cleanup_stage_db

(Optional) Use this parameter to clean the staging tables before using the Stage Copy utility. When you use the `-cleanup_stage_db` parameter to clean the site data, note that the merchant data can be deleted because of the delete cascade. You should clean and copy the merchant data after you clean and copy the site data. Yes is the default. If you specify no, nothing will be deleted from the staging tables. Your copy might fail if your copy data generates conflict or duplicate key on primary key or unique indexes. To use the staging copy to clean up your stage database only, without a data copy from the production database, specify the `-cleanup_stage_db` as only.

script file

(Optional) The name of the SQL script file generated by the Stage Copy utility command when using export and import to copy the production database to the staging database on the specified scope. The script file also generates the delete statements to clean the staging database if you use the default value or specify `-cleanup_stage_db` as yes.

Note: This script file is not supported on iSeries.

Before you run the script, verify that you have enough disk space to hold the exported tables. The script file is located in the Stage Copy utility directory where you invoke the Stage Copy utility.

 Use `db2 -vtd# -f script_file_name` to run the script file.

dbtype (Optional) The database type (DB2 or Oracle). The default is DB2.







For more information on copying the staging server, see the examples.

Examples of copying data to the staging database: The following examples illustrate how you can copy tables from the production database to the staging database. It is important to remember that you cannot use the Stage Copy utility if RFQs are on your production system. For details, see “Staging server limitations” on page 34.

Note that you should type the entire command in a single line. The commands are shown here on more than one line for presentation purposes only.



Example 1



After cleaning the staging database, copy the production database to the staging database with the scope set to all:






1. Set the PATH environment variables.   Not applicable.
2. Configure the database.   Not applicable.
3. Change to the directory to which you want log files written.   Not applicable. For iSeries the log files will default to




```
/QIBM/UserData/CommerceServer55/instances/stagingcopy_{sourcedb_user}  
_{destdb_user}_{timestamp}.log)
```

4. Type the following:

-  
stagingcopy -scope _all_ -sourcedb *production_database_name* -destdb
staging_database_name

-  
stagingcopy -scope _all_ -sourcedb *production_database_name* -destdb
staging_database_name-dbtype oracle -sourcedb_user *user*
-sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*



-     
. stagingcopy.sh -scope _all_ -sourcedb *production_database_name*
-destdb *staging_database_name* -sourcedb_user *user* -destdb_user *user*



-   
. stagingcopy.sh -scope _all_ -sourcedb *production_database_name*
-destdb *staging_database_name* dbtype oracle -sourcedb_user *user*
-sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*






Note: Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.




5. Examine the stagingcopy_YYYY.MM.DD_HH.MM.SS.ZZZ.log file to verify that the command was successful.

To specify the log file name and path, use the log file parameter:

-  
stagingcopy -scope _all_ -sourcedb *production_database_name* -destdb
staging_database_name -log *log_file_name*

-  
stagingcopy -scope _all_ -sourcedb *production_database_name* -destdb
staging_database_name -log *log_file_name* -dbtype oracle -sourcedb_user
user -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*

-     

```
. stagingcopy.sh -scope _all_ -sourcedb production_database_name -destdb
staging_database_name -log log_file_name -sourcedb_user user -destdb_user
user
```
-   

```
. stagingcopy.sh -scope _all_ -sourcedb
production_database_name -destdb staging_database_name -log log_file_name
dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user
user -destdb_passwd password
```



Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.



If you are using DB2 and are not logged on as the Database Administrator, you need to provide values for the `-sourcedb_user`, `-sourcedb_passwd`, `-destdb_user`, and `-destdb_passwd` options.






Example 2




After cleaning the merchant tables from staging database, copy the merchant-related tables from the production database to staging database:

1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written.
4. Type the following:

-  

```
stagingcopy -scope _merchant_ -sourcedb
production_database_name -destdb staging_database_name
```
-  

```
stagingcopy -scope _merchant_ -sourcedb
production_database_name -destdb staging_database_name -dbtype oracle
-sourcedb_user user -sourcedb_passwd password -destdb_user user
-destdb_passwd password
```
-     



```
stagingcopy.sh -scope
_merchant_ -sourcedb production_database_name -destdb
staging_database_name -sourcedb_user user -destdb_user user
```
-   



```
stagingcopy.sh -scope _merchant_ -sourcedb
production_database_name -destdb staging_database_name -dbtype oracle
-sourcedb_user user -sourcedb_passwd password -destdb_user user
-destdb_passwd password
```

Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

5. Examine the `stagingcopy_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

To only clean the staging database, specify the `-cleanup_stage_db` parameter:

-  

```
stagingcopy -scope _merchant_ -sourcedb
production_database_name -destdb staging_database_name -cleanup_stage_db
only
```
-  

```
stagingcopy -scope _merchant_ -sourcedb
production_database_name -destdb staging_database_name -cleanup_stage_db
only -dbtype oracle -sourcedb_user user -sourcedb_passwd password
-destdb_user user -destdb_passwd password
```

- ▶ AIX
▶ 400
▶ Solaris
▶ Linux
▶ DB2
 stagingcopy.sh -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db only -sourcedb_user *user* -destdb_user *user*
- ▶ AIX
▶ Solaris
▶ Oracle
 stagingcopy.sh -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db only dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*

Note: Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

To only copy data, specify the -cleanup_stage_db no parameter:

- ▶ Windows
▶ DB2
 stagingcopy -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no
- ▶ Windows
▶ Oracle
 stagingcopy -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*
- ▶ AIX
▶ 400
▶ Solaris
▶ Linux
▶ DB2
 stagingcopy.sh -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -sourcedb_user *user* -destdb_user *user*
- ▶ AIX
▶ Solaris
▶ Oracle
 stagingcopy.sh -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*




Note: Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

Important: When copying with the scope set to merchant, ensure that you have copied the site scope data first. Otherwise, your copy will fail.

Example 3

After cleaning the site tables from staging database, copy the site tables from production database to stage database.



1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written
4. Type the following:
 - ▶ Windows
▶ DB2
 stagingcopy -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name*
 - ▶ Windows
▶ Oracle
 stagingcopy -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name* -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*
 - ▶ AIX
▶ 400
▶ Solaris
▶ Linux
▶ DB2
 stagingcopy.sh -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name* -sourcedb_user *user* -destdb_user *user*



-    `stagingcopy.sh -scope _site_ -sourcedb production_database_name -destdb staging_database_name -dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user user -destdb_passwd password`






Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.




5. Examine the `stagingcopy_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

Deleting the site tables may impact the merchant tables due to the delete cascade restriction. Clean the merchant data first, followed by the site data, and then copy the data:



-   `stagingcopy -scope _merchant_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only`



-   `stagingcopy -scope _merchant_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only -dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user user -destdb_passwd password`






-      `stagingcopy.sh -scope _merchant_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only -sourcedb_user user -destdb_user user`




-    `stagingcopy.sh -scope _merchant_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only -dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user user -destdb_passwd password`

Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.













-   `stagingcopy -scope _site_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only`

-   `stagingcopy -scope _site_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only -dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user user -destdb_passwd password`













-      `stagingcopy.sh -scope _site_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only -sourcedb_user user -destdb_user user`

-    `stagingcopy.sh -scope _site_ -sourcedb production_database_name -destdb staging_database_name -cleanup_stage_db only -dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user user -destdb_passwd password`

Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.


-   stagingcopy -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no
-   stagingcopy -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*
-      stagingcopy.sh -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -sourcedb_user *user* -destdb_user *user*
-    stagingcopy.sh -scope _site_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*

Note: Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.





-   stagingcopy -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no
-   stagingcopy -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*
-      stagingcopy.sh -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -sourcedb_user *user* -destdb_user *user*
-    stagingcopy.sh -scope _merchant_ -sourcedb *production_database_name* -destdb *staging_database_name* -cleanup_stage_db no -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*








Note: Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.


Example 4

Generate the following script to clean and copy the production database to the stage database with scope all.  This example does not apply to OS/400 for iSeries, since the -script option is NOT supported on iSeries.

1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written.
4. Type the following:

-   stagingcopy -scope _all_ -sourcedb *production_database_name* -destdb *staging_database_name* -script_file *stage_copy.sql*
-   stagingcopy -scope _all_ -sourcedb *production_database_name* -destdb *staging_database_name* -script_file *stage_copy.sql* -dbtype oracle -sourcedb_user *user* -sourcedb_passwd *password* -destdb_user *user* -destdb_passwd *password*

-     stagingcopy.sh -scope_all_ -sourcedb production_database_name -destdb staging_database_name -script_file stage_copy.sql -sourcedb_user user -destdb_user user
 -    stagingcopy.sh -scope_all_ -sourcedb production_database_name -destdb staging_database_name -script_file stage_copy.sql dbtype oracle -sourcedb_user user -sourcedb_passwd password -destdb_user user -destdb_passwd password
- Note:** Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.
5. Examine the stagingcopy_yyyy.mm.dd_hh.mm.ss.zzz.log file to verify that the command was successful.

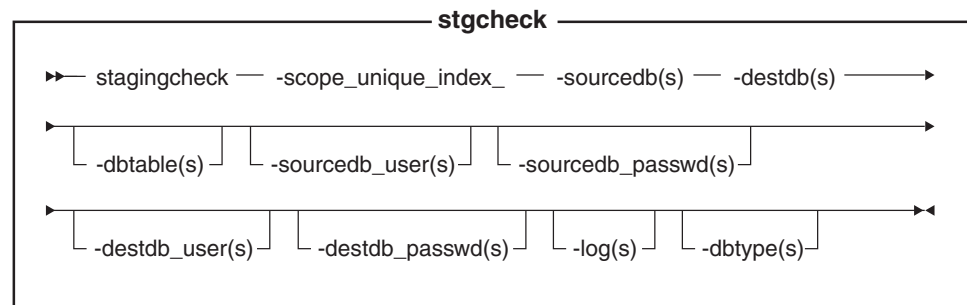
The Stage Copy utility generates the stage_copy.sql script to clean and copy the database.  If you are using DB2, run the following script:


1. Logon as the Database Administrator (DBA).
2. Open a DB2 command window.
3. Type: db2 -vtd# -f stage_copy.sql

 **Oracle** If you are using Oracle, run the following script:

1. Open an SQLPlus window.
2. Connect as dba.
3. Type: @stage_copy.sql

Stage Check command (Windows 2000, AIX, iSeries, Linux, and Solaris): The Stage Check command determines if there is a unique index key conflict between the staging database and the production database. To run the Stage Check utility, type the following from a command line on staging server or production server. Type the entire command on one line. It is shown here on more than one line for presentation purpose only.



 **Note:** You must include the optional parameters, logon user ID and password in the command even if you are currently running this utility with the same user ID.

 **400** iSeries now uses stagingCheck.sh. To run shell scripts on iSeries do the following:

1. Log on as a user profile that has a CCSID other than 65535.
2. Open a QSHLL command window by typing the following command on an OS/400 command line: STRQSH.
3. Run the command as follows:
/QIBM/ProdData/CommerceServer55/bin/stagingCheck.sh (parameters . . .)

Note: The user parameters must always be specified.

z/OS If you are working with DB2 for z/OS, you need to provide some additional parameters relative to the above syntax:

- -dbtype DB2/390
- -sourcedb_user *Source_DB_user*
- -sourcedb_password *Source_DB_password*
- -sourcedb_schema *Source_DB_schema* (for DB2 for z/OS databases only)
- -destdb_user *Destination_DB_user*
- -destdb_password *Destination_DB_password*
- -destdb_schema *Destination_DB_schema* (for DB2 for z/OS databases only)

Note the following:

- You need to specify the following parameters that are exclusively for Linux Hybrid:
 - -sourcedb_schema
 - -destdb_schema
- Both users (specified by -sourcedb_user and -destdb_user) need to have the DBADM privilege on the database objects created by each other
- The following parameters are required (and not optional as on other platforms):
 - -sourcedb_user
 - -sourcedb_passwd
 - -destdb_user
 - -destdb_passwd
- The DB type specified by -dbtype must be DB2/390

Parameter values

scope The level of scope for the copy to the staging server. Specify `_unique_index_`.

sourcedb

The name of the database on the staging server.

Oracle Use `host:port:sid`. For example, `myhost:1521:mydb`.

destdb The name of the database on the production server.

Oracle Use `host:port:sid`. For example, `myhost:1521:mydb`.

dbtable

(Optional) The name of a specific table to be checked for unique key conflicts.

sourcedb_user

(Optional) The logon ID of the database administrator who has created the schema of the staging database. If not specified, the ID of the user currently invoking the utility is used.

400 (Required) The user profile associated with the commerce instance. This is the same as the source database schema.

sourcedb_passwd

(Optional) The password of the logon ID that is specified by the sourcedb parameter.

destdb_user

(Optional) The logon ID of the database administrator who has created the

schema of the production database. If not specified, the ID of the user invoking the utility is used.

▶ 400 (Required) The user profile associated with the commerce instance. This is the same as the destination database schema.

destdb_passwd

(Optional) The password of the logon ID that is specified by the `destdb_user` parameter. If not specified, the system prompts you to enter the password.

log

(Optional) The path and name of the file in which the Stage Copy utility records its activities and errors. If this parameter is not specified, a log file called `stagingcopy_yyyy.mm.dd_hh.mm.ss.zzz.log` is created in the following log directory.

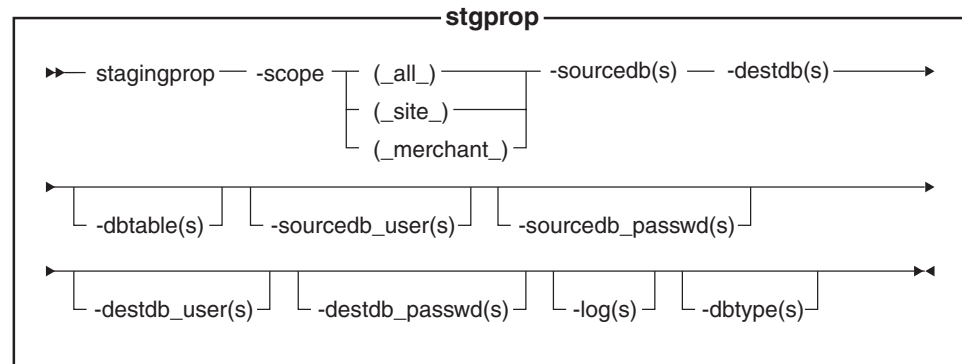
- ▶ AIX ▶ Solaris ▶ Linux `WC_installdir/logs`
- ▶ Windows `WC_installdir\logs`
- ▶ 400 `WC_userdir/instances.`

dbtype (Optional) The database type (DB2 or Oracle). The default is DB2.

For more information on checking for unique index key conflicts, see the examples.

Stage Propagate utility command (Windows 2000, AIX, iSeries, Linux, and Solaris): The Stage Propagate utility copies data from the staging database to the production database. Note that you cannot create or update RFQ objects on the staging server.

To run the Stage Propagate utility, type the following from a command line on a machine that can connect to both the staging server and the production server database. Type the entire command on one line. It is shown here on more than one line for presentation purposes only.



▶ Oracle

Note: You must include the optional parameters, logon user ID and password in the command even if you are currently running this utility with the same user ID.

▶ 400 iSeries now uses `stagingProp.sh`. To run shell scripts on iSeries do the following:

1. Log on as a user profile that has a CCSID other than 65535.
2. Open a QSHLL command window by typing the following command on an OS/400 command line: `STRQSH`.
3. Run the command as follows:

/QIBM/ProdData/CommerceServer55/bin/stagingProp.sh (*parameters . . .*)

Note: The user parameters must always be specified.

z/OS If you are working with DB2 for z/OS, you need to provide some additional parameters relative to the above syntax:

- -dbtype DB2/390
- -sourcedb_user *Source_DB_user*
- -sourcedb_password *Source_DB_password*
- -sourcedb_schema *Source_DB_schema* (for DB2 for z/OS databases only)
- -destdb_user *Destination_DB_user*
- -destdb_password *Destination_DB_password*
- -destdb_schema *Destination_DB_schema* (for DB2 for z/OS databases only)

Note the following:

- You need to specify the following parameters that are exclusively for Linux Hybrid:
 - -sourcedb_schema
 - -destdb_schema
- Both users (specified by -sourcedb_user and -destdb_user) need to have the DBADM privilege on the database objects created by each other
- The following parameters are required (and not optional as on other platforms):
 - -sourcedb_user
 - -sourcedb_passwd
 - -destdb_user
 - -destdb_passwd
- The DB type specified by -dbtype must be DB2/390

Parameter values

scope

The scope level for the propagation to the production server. Specify one of the following:

- `_all_`
Type `_all_` to propagate both record related to site and to all merchants.
- `_site_`
Type `_site_` to propagate only site-related record.
- `_merchant_`
Type `_merchant_` to propagate only records related to all merchants.

sourcedb

The name of the database on the staging server. **Oracle** Use `host:port:sid`. For example, `myhost:1521:mydb`.

destdb The name of the database on the production server.

Oracle Use `host:port:sid`. For example, `myhost:1521:mydb`.

dbtable

(Optional) The name of any specific table to be propagated. All changed records in this table will be propagated, provided the records are within the scope specified by the scope parameter; otherwise, no records will be propagated.

sourcedb_user

(Optional) The logon ID of the database administrator who has created the source database schema. If not specified, the ID of the user currently invoking the utility is used.

 (Required) The user profile associated with the commerce instance. This is the same as the source database schema.

sourcedb_passwd

(Optional) The password of the logon ID that is specified by the sourcedb_user parameter.

destdb_user

(Optional) The logon ID of the database administrator who has created the destination database schema. If not specified, the ID of the user invoking the utility is used. This parameter is *mandatory* when using a remote database.






 (Required) The user profile associated with the commerce instance. This is the same as the destination database schema.

destdb_passwd

(Optional) The password of the logon ID that is specified by the destdb_user parameter. If not specified, the system prompts you to enter the password. This parameter is mandatory when using a remote database.

log

(Optional) The path and name of the file in which the Stage Propagate utility records its activities and errors. If this parameter is not specified, a log file called `stagingprop_yyyy.mm.dd_hh.mm.ss.zzz.log` is created in the following log directory.

   `WC_installdir/logs`
 `WC_installdir\logs`
 `WC_userdir/instances`

dbtype (Optional) The database type (DB2 or Oracle). The default is DB2.

For more information on propagating to the staging server, see the following examples.





Examples of propagating data to the production database: The following examples illustrate how you propagate changed records from a staging database to a production database.

Note that you should type the commands in a single line. The commands are shown here on more than one line for presentation purposes only.

Example 1

Propagate all changes from the staging server database to the production database.

1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written
4. Type the following:

-   `stagingprop -scope_all_ -sourcedb staging_database_name -destdb production_database_name`
-   `stagingprop -scope_all_ -sourcedb staging_database_name -destdb production_database_name`

```
-dbtype oracle -sourcedb_user user -sourcedb_passwd password
-destdb_user user -destdb_passwd password
```

- ```

AIX 400 Solaris Linux DB2 stagingprop.sh -scope
all -sourcedb staging_database_name -destdb production_database_name
-sourcedb_user user -destdb_user user

```
- ```

AIX Solaris Oracle stagingprop.sh -scope _all_
-sourcedb staging_database_name -destdb production_database_name
dbtype oracle -sourcedb_user user -sourcedb_passwd password
-destdb_user user -destdb_passwd password

```

Note: Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

- Examine the log file to verify that the command was successful. Check the description of the logs parameters to determine the default location of this log file for your database type and platform.

The Stage Propagate utility will first propagate all site data, and then all merchant data to the production database. If an error occurs, the entire transaction will rollback.

To specify the log file name and path, use the log file parameter:

- ```

Windows DB2 stagingprop -scope _all_
-sourcedb staging_database_name -destdb production_database_name
-loglog_file_name

```
- ```

Windows Oracle stagingprop -scope _all_
-sourcedb staging_database_name -destdb production_database_name
-loglog_file_name -dbtype oracle -sourcedb_user user -sourcedb_passwd
password -destdb_user user -destdb_passwd password

```
- ```

AIX 400 Solaris Linux DB2 stagingprop.sh -scope _all_
-sourcedb staging_database_name -destdb production_database_name
-loglog_file_name -sourcedb_user user -destdb_user user

```
- ```

AIX Solaris Oracle
stagingprop.sh -scope _all_ -sourcedb staging_database_name -destdb
production_database_name -loglog_file_name -dbtype oracle -sourcedb_user
user -sourcedb_passwd password -destdb_user user -destdb_passwd password

```

Note: Use *host:port:sid* for the Oracle database name, for example, myhost:1521:mydb.

If you are using DB2 and are not logged on as the Database Administrator, you need to provide values for the `-sourcedb_user`, `-sourcedb_passwd`, `-destdb_user`, and `-destdb_passwd` options.











Example 2

Propagate all modified site data from the staging database to the production database.

- Set the PATH environment variables.
- Configure the database.
- Change to the directory to which you want log files written
- Type the following:
 - ```

Windows DB2 stagingprop -scope _site_
-sourcedb staging_database_name -destdb production_database_name

```

-   stagingprop -scope \_site\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name* -dbtype oracle -sourcedb\_user *user* -sourcedb\_passwd *password* -destdb\_user *user* -destdb\_passwd *password*
-      stagingprop.sh -scope \_site\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name* -sourcedb\_user *user* -destdb\_user *user*
-    stagingprop.sh -scope \_site\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name* dbtype oracle -sourcedb\_user *user* -sourcedb\_passwd *password* -destdb\_user *user* -destdb\_passwd *password*













**Note:** Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

5. Examine the log file to verify that the command was successful. Check the description of the logs parameters to determine the default location of this log file for your database type and platform.

### Example 3

Propagate all modified merchant data from the staging database to production database (after propagating the site data.)

1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written
4. Type the following:

-   stagingprop -scope \_merchant\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name*
-   stagingprop -scope \_merchant\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name* -dbtype oracle -sourcedb\_user *user* -sourcedb\_passwd *password* -destdb\_user *user* -destdb\_passwd *password*
-      stagingprop.sh -scope \_merchant\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name* -sourcedb\_user *user* -destdb\_user *user*
-    stagingprop.sh -scope \_merchant\_ -sourcedb *staging\_database\_name* -destdb *production\_database\_name* dbtype oracle -sourcedb\_user *user* -sourcedb\_passwd *password* -destdb\_user *user* -destdb\_passwd *password*

**Note:** Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.


5. Examine the log file to verify that the command was successful. Check the description of the logs parameters to determine the default location of this log file for your database type and platform.

### Customized database table requirements

If you have customized your database schema by creating new tables, you must meet the following requirements to use the staging server:

- **You must define a primary key or a unique index.**  
The staging server functions based on the key. To avoid logging excessive data in the STAGLOG table, log only the key (primary key or unique index). The

stage utilities will use the key for compression and to find the data to be propagated. If there is no key, the stage utilities cannot work.

- **A referential integrity (RI) constraint cycle cannot exist among the tables.**  
The staging server always propagates the parent table before the child table. If there is an RI constraint cycle, the staging server cannot distinguish between parent and child tables.
- **The database tables only contain configuration data.**  
In a business-to-consumer scenario, configuration data is under Site Administrator control, such as catalogs and catalog entries. If a table contains operation data, a customer can change the same table in a production database after a Site Administrator has copied the table to the staging database. This causes a potential key conflict or an RI constraint violation.
- **The database tables cannot contain any references to operation tables.**  
The tables to be propagated should not contain any foreign key references to the primary keys of operation tables. If there is such a reference, the data cannot be restored to the product database if a customer deletes the primary key after the stage copy.
-  **An insert trigger cannot exist when inserting two tables in the production database.**  
For any two tables covered by the staging server (for example, R1 and R2), a trigger to insert rows into R1 or R2 cannot exist when inserting into R2 and R1 in the production database. The insert trigger creates the update in both databases and generates key problems.
- **The MEMBER table cannot have a unique index.**
- **Delete restrict on the customized database tables must be used with caution.**  
Delete restrict inhibits the Database Cleanup utility's performance. You can also experience difficulties when cleaning the staging database. Before you can clean the staging database, you have to manually use the Database Cleanup utility command with the force option to clean the tables. Otherwise, cleaning the staging database will fail.

To prepare the staging server for customized tables, refer to configuring the staging server for customized tables.

## Staging server troubleshooting (Windows 2000, AIX, Linux, and Solaris)

1. While using the staging server command `stgprop`, you may receive the following message:  
SQLSTATE 54001: The statement is too long or too complex.

Ensure that you have set the `stmheap` size of your database as suggested in configuring the database. You should verify that you have enough memory to support the increase.

2. If the Stage Copy utility fails to complete processing, the database log may be too small. From a DB2 command window issue the commands:  

```
update database configuration for staging_server using logprimary 50
logfilsiz 1000
db2 terminate
db2stop
db2start
```

where *staging\_server* is the name of the staging server database. If the problem persists, try using a higher value for the `logprimary` or `logfilsiz` parameters.

3. **AIX** Ensure that the resource for DB2 user, such as db2inst1, has been configured properly. Entering the following command shows your resource limitation: `ulimit -a`  
 Configure the data segment to 240MB and stack to 16MB.  
 For additional information, refer to the *DB2 Command Reference*.

### WebSphere Commerce staging tables

You can copy WebSphere Commerce staging tables from the production server to the staging server. The tables are grouped according to whether they contain site-related data, merchant-related data, or both site and merchant-related data.

Each table group is listed at the following pages:

- Site data scope
- Merchant data scope
- Site and merchant data scope

### Triggers for staging tables

The following triggers have been defined for the WebSphere Commerce staging tables. You can apply these settings to custom tables if they contain the same data scope and key characteristics. Refer to the links below to view the tables by data scope.

- Site data scope
- Merchant data scope
- Site or merchant data scope

**Site data scope:** The following chart lists the database tables under the site data scope.

| Table name | Unique index on |           |        |           |        |
|------------|-----------------|-----------|--------|-----------|--------|
|            | Integer 1       | Integer 2 | Char 1 | Integer 3 | Char 2 |
| ACACGPDESC | Yes             | Yes       |        |           |        |
| ACACTACTGP | Yes             | Yes       |        |           |        |
| ACACTDESC  | Yes             | Yes       |        |           |        |
| ACACTGRP   | Yes             |           |        |           |        |
| ACACTION   | Yes             |           |        |           |        |
| ACATTR     | Yes             |           |        |           |        |
| ACATTRDESC | Yes             | Yes       |        |           |        |
| ACCCMDTYPE |                 |           | Yes    |           |        |
| ACRELATION | Yes             |           |        |           |        |
| ACRELDESC  | Yes             | Yes       |        |           |        |
| ACRESACT   | Yes             | Yes       |        |           |        |
| ACRESATREL | Yes             | Yes       |        |           |        |
| ACRESCGRY  | Yes             |           |        |           |        |
| ACRESGRP   | Yes             |           |        |           |        |
| ACRESGPDES | Yes             | Yes       |        |           |        |
| ACRESGPRES | Yes             | Yes       |        |           |        |
| ACRESPRIM  | Yes             |           |        |           |        |
| ACRESREL   | Yes             | Yes       |        |           |        |

|            |     |     |     |     |     |
|------------|-----|-----|-----|-----|-----|
| ATTACHUSG  |     |     | Yes |     |     |
| ATTRTYPE   |     |     | Yes |     |     |
| BUYERPOTYP | Yes |     |     |     |     |
| CALUSAGE   | Yes |     |     |     |     |
| CATENTTYPE |     |     | Yes |     |     |
| CATRELTYPE |     |     | Yes |     |     |
| CHKARRANG  | Yes | Yes |     |     |     |
| CHKCMD     | Yes |     |     |     |     |
| COUNTCODE  |     |     | Yes |     | Yes |
| COUNTRY    | Yes |     | Yes |     |     |
| DEVICEFMT  | Yes |     |     |     |     |
| FLCOMPOSE  | Yes | Yes |     | Yes |     |
| FLDOMNDESC | Yes | Yes |     |     |     |
| FLOW       | Yes |     |     |     |     |
| FLOWDESC   | Yes | Yes |     |     |     |
| FLOWDOMAIN | Yes |     |     |     |     |
| FLOWTYPE   | Yes |     |     |     |     |
| FLSTATEDCT | Yes |     |     |     |     |
| FLSTATEGP  | Yes |     |     |     |     |
| FLSTATEREL | Yes | Yes |     |     |     |
| FLSTDCTDSC | Yes | Yes |     |     |     |
| FLSTGPDSC  | Yes | Yes |     |     |     |
| FLTRANSDSC | Yes | Yes |     |     |     |
| FLTRANSITN | Yes |     |     |     |     |
| FLTYPEDESC | Yes | Yes |     |     |     |
| ICDATAREG  | Yes |     |     |     |     |
| INVRSRVTYP | Yes |     |     |     |     |
| INVRSRVDSC | Yes | Yes |     |     |     |
| ITEMTYPE   |     |     | Yes |     |     |
| LANGUAGE   | Yes |     |     |     |     |
| LANGUAGEDS | Yes | Yes |     |     |     |
| MASSOC     |     |     | Yes |     |     |
| MASSOCTYPE |     |     | Yes |     |     |
| MBRATTR    | Yes |     |     |     |     |
| MBRGRPTYPE | Yes |     |     |     |     |
| OPERATOR   | Yes |     |     |     |     |
| OPERATRDSC | Yes | Yes |     |     |     |
| ORDCHNLTY  | Yes |     |     |     |     |
| OUTPUTQ    | Yes |     |     |     |     |
| OUTPUTQDSC | Yes | Yes |     |     |     |
| PARTROLE   | Yes |     |     |     |     |
| PARTROLEDS | Yes | Yes |     |     |     |



|            |     |     |     |  |     |
|------------|-----|-----|-----|--|-----|
| PLCYTYPDSC | Yes |     | Yes |  |     |
| POLICYTYPE | Yes |     |     |  |     |
| QTYCONVERT |     |     | Yes |  | Yes |
| QTYUNIT    |     |     | Yes |  |     |
| QTYUNITDSC | Yes |     | Yes |  |     |
| ROLE       | Yes |     |     |  |     |
| SCHCMD     | Yes |     |     |  |     |
| SETCURR    |     |     | Yes |  |     |
| SETCURRDSC | Yes |     | Yes |  |     |
| STATECODE  |     |     | Yes |  | Yes |
| STATEPROV  | Yes |     | Yes |  |     |
| STORECGRY  | Yes |     |     |  |     |
| TAXTYPE    | Yes |     |     |  |     |
| TCSBTYPDS  | Yes |     | Yes |  |     |
| TCSBTYPPE  |     |     | Yes |  |     |
| TCTYPE     |     |     | Yes |  |     |
| TFALGOPOL  | Yes |     |     |  |     |
| TFALGOTYPE | Yes |     |     |  |     |
| TFALGPOLDS | Yes | Yes |     |  |     |
| TFALGTYPDS | Yes | Yes |     |  |     |
| TFDOMAIN   | Yes |     |     |  |     |
| TFDOMDSC   | Yes | Yes |     |  |     |
| TFSBDOMAIN | Yes |     |     |  |     |
| TFSBDOMDSC | Yes | Yes |     |  |     |
| TRDTYPE    | Yes |     |     |  |     |
| TRDTYPEDSC | Yes | Yes |     |  |     |
| TXCDScheme | Yes |     |     |  |     |

**Merchant data scope:** The following chart lists the database tables under the merchant data scope.

| Table name | Unique index on |           |        |           |        |
|------------|-----------------|-----------|--------|-----------|--------|
|            | Integer 1       | Integer 2 | Char 1 | Integer 3 | Char 2 |
| ACCCMDGRP  | Yes             |           |        |           |        |
| ACCCUSTEXC | Yes             | Yes       |        |           |        |
| ACCOUNT    | Yes             |           |        |           |        |
| ACORGPOL   | Yes             | Yes       |        |           |        |
| ACPOLDESC  | Yes             | Yes       |        |           |        |
| ACPOLICY   | Yes             |           |        |           |        |
| ATTACHMENT | Yes             |           |        |           |        |
| ATTRIBUTE  | Yes             | Yes       |        |           |        |
| ATTRVALUE  | Yes             | Yes       |        |           |        |

|            |     |     |     |     |     |
|------------|-----|-----|-----|-----|-----|
| BASEITEM   | Yes |     |     |     |     |
| BASEITEM   | Yes | Yes |     |     |     |
| BUYERPO    | Yes |     |     |     |     |
| CALCODE    | Yes |     |     |     |     |
| CALCODEDSC | Yes | Yes |     |     |     |
| CALCODEMGP | Yes | Yes |     |     |     |
| CALCOTXEX  | Yes | Yes |     |     |     |
| CALMETHOD  | Yes |     |     |     |     |
| CALRANGE   | Yes |     |     |     |     |
| CALRLOOKUP | Yes |     |     |     |     |
| CALRULE    | Yes |     |     |     |     |
| CALRULEMGP | Yes | Yes |     |     |     |
| CALSCALE   | Yes |     |     |     |     |
| CALSCALEDS | Yes | Yes |     |     |     |
| CATALOG    | Yes |     |     |     |     |
| CATALOGDSC | Yes | Yes |     |     |     |
| CATCNTR    | Yes | Yes |     |     |     |
| CATCONFINF | Yes |     |     |     |     |
| CATENCALCD | Yes |     |     |     |     |
| CATENTATTR | Yes |     |     |     |     |
| CATENTDESC | Yes | Yes |     |     |     |
| CATENTREL  | Yes | Yes | Yes |     |     |
| CATENTRY   | Yes |     |     |     |     |
| CATENTSHIP | Yes |     |     |     |     |
| CATGPCALCD | Yes |     |     |     |     |
| CATGPENREL | Yes | Yes |     | Yes |     |
| CATGROUP   | Yes |     |     |     |     |
| CATGRPATTR | Yes | Yes |     |     |     |
| CATGRPDESC | Yes | Yes |     |     |     |
| CATGRPPS   | Yes | Yes |     | Yes |     |
| CATGRPREL  | Yes | Yes |     | Yes |     |
| CATGRPTPC  | Yes | Yes |     | Yes |     |
| CATTOGRP   | Yes | Yes |     |     |     |
| CHARGETYPE | Yes |     |     |     |     |
| CHRGTYPDSC | Yes | Yes |     |     |     |
| CNTRDISPLY | Yes |     |     |     |     |
| CNTRNAME   | Yes |     | Yes |     |     |
| CONTRACT   | Yes |     |     |     |     |
| CREDITLINE | Yes |     |     |     |     |
| CRULESCALE | Yes | Yes |     |     |     |
| CURCONVERT | Yes |     |     |     |     |
| CURCVLIST  | Yes | Yes | Yes |     | Yes |

|            |     |     |     |     |  |
|------------|-----|-----|-----|-----|--|
| CURFMTDESC | Yes | Yes | Yes |     |  |
| CURFORMAT  | Yes |     | Yes |     |  |
| CURLIST    | Yes |     | Yes |     |  |
| DISPCGPREL | Yes |     |     |     |  |
| DISPENTREL | Yes |     |     |     |  |
| DISTARRANG | Yes |     |     |     |  |
| FLOWADMIN  | Yes |     |     |     |  |
| INVADJCODE | Yes |     |     |     |  |
| INVADJDESC | Yes | Yes |     |     |  |
| ITEMSPC    | Yes |     |     |     |  |
| ITEMVERSN  | Yes |     |     |     |  |
| JURST      | Yes |     |     |     |  |
| JURSTGPREL | Yes | Yes |     |     |  |
| JURSTGROUP | Yes |     |     |     |  |
| LANGPAIR   | Yes | Yes |     | Yes |  |
| LISTPRICE  | Yes |     | Yes |     |  |
| MASSOCCECE | Yes |     |     |     |  |
| MASSOCGPGP | Yes |     |     |     |  |
| MBRATTRVAL | Yes |     |     |     |  |
| MBRGRPUSG  | Yes | Yes |     |     |  |
| MBRREL     | Yes | Yes |     |     |  |
| MBRROLE    | Yes | Yes |     | Yes |  |
| MGPTRDPSCN | Yes | Yes |     |     |  |
| OFFER      | Yes |     |     |     |  |
| OFFERDESC  | Yes | Yes |     |     |  |
| OFFERPRICE | Yes |     | Yes |     |  |
| PARTICIPNT | Yes |     |     |     |  |
| PATTRDESC  | Yes | Yes |     |     |  |
| PATTRIBUTE | Yes |     |     |     |  |
| PATTRPROD  | Yes | Yes |     |     |  |
| PKGATTR    | Yes |     |     |     |  |
| PKGATTRVAL | Yes |     |     |     |  |
| PKGITEMREL | Yes | Yes |     | Yes |  |
| POLICY     | Yes |     |     |     |  |
| POLICYCMD  | Yes |     | Yes |     |  |
| POLICYDESC | Yes | Yes |     |     |  |
| POLICYTC   | Yes | Yes |     |     |  |
| PRODSETDSC | Yes | Yes |     |     |  |
| PRODUCTSET | Yes |     |     |     |  |
| PRSETCEREL | Yes | Yes |     |     |  |
| QTYFMTDESC | Yes | Yes | Yes |     |  |
| QTYFORMAT  | Yes |     | Yes |     |  |

|            |     |     |     |     |  |
|------------|-----|-----|-----|-----|--|
| RTNDNYDESC | Yes | Yes |     |     |  |
| RTNDNYRSN  | Yes |     |     |     |  |
| RTNDSPCODE | Yes |     |     |     |  |
| RTNDSPDESC | Yes | Yes |     |     |  |
| RTNREASON  | Yes |     |     |     |  |
| RTNRSNDESC | Yes | Yes |     |     |  |
| SHPARJURGP | Yes | Yes |     |     |  |
| SHPARRANGE | Yes |     |     |     |  |
| SHPJCRULE  | Yes |     |     |     |  |
| SHPMODEDSC | Yes | Yes |     |     |  |
| STENCALUSG | Yes | Yes |     |     |  |
| STORECAT   | Yes | Yes |     |     |  |
| STORECENT  | Yes | Yes |     |     |  |
| STORECGRP  | Yes | Yes |     |     |  |
| STORECNTR  | Yes | Yes |     |     |  |
| STOREDEF   | Yes |     |     |     |  |
| STOREITEM  | Yes | Yes |     |     |  |
| STORELANG  | Yes | Yes |     |     |  |
| STOREMBRGP | Yes | Yes |     |     |  |
| STORITMFFC | Yes | Yes |     | Yes |  |
| STORLANGDS | Yes | Yes |     | Yes |  |
| TAXCGRY    | Yes |     |     |     |  |
| TAXCGRYDS  | Yes | Yes |     |     |  |
| TAXJCRULE  | Yes | Yes |     | Yes |  |
| TCDESC     | Yes | Yes |     |     |  |
| TDPSCNCNTR | Yes | Yes |     |     |  |
| TERMCOND   | Yes |     |     |     |  |
| TRADEPOSCN | Yes |     |     |     |  |
| TRADING    | Yes |     |     |     |  |
| TRDATTACH  | Yes | Yes |     |     |  |
| TRDDESC    | Yes | Yes |     |     |  |
| TXCDCLASS  | Yes |     |     |     |  |
| VENDOR     | Yes |     |     |     |  |
| VENDORDESC | Yes | Yes |     |     |  |
| VERSIONSPC | Yes |     |     |     |  |
| VIEWREG    | Yes | Yes | Yes |     |  |

**Site and merchant data scope:** The following chart lists the database tables under the site and merchant data scope.

| Table name | Unique index on |           |        |           |        |
|------------|-----------------|-----------|--------|-----------|--------|
|            | Integer 1       | Integer 2 | Char 1 | Integer 3 | Char 2 |

|            |     |     |     |  |  |
|------------|-----|-----|-----|--|--|
| CMDREG     | Yes |     | Yes |  |  |
| FFMCENTDS  | Yes | Yes |     |  |  |
| FFMCENTER  | Yes |     |     |  |  |
| MBRGRP     | Yes |     |     |  |  |
| MBRGRPCOND | Yes |     |     |  |  |
| MEMBER     | Yes |     |     |  |  |
| ORGENTITY  | Yes |     |     |  |  |
| SHIPMODE   | Yes |     |     |  |  |
| STADDRESS  | Yes |     |     |  |  |
| STORE      | Yes |     |     |  |  |
| STOREENT   | Yes |     |     |  |  |
| STOREENTDS | Yes | Yes |     |  |  |
| STOREGRP   | Yes |     |     |  |  |
| URLREG     | Yes |     | Yes |  |  |
| VIEWREG    | Yes | Yes | Yes |  |  |

## Connectivity services

WebSphere Commerce uses the Program Adapter for HTTP, Adapter for WebSphere MQ, and Adapter for CrossWorlds, for connectivity to external systems. The following sections describe each adapter and how they are used in WebSphere Commerce.

### Program Adapter for HTTP

The Program Adapter for HTTP allows external systems to communicate with WebSphere Commerce by passing XML requests over the HTTP protocol. The Program Adapter for HTTP provides external systems such as procurement systems with a common way to communicate with WebSphere Commerce through HTTP, allowing WebSphere Commerce to act as a supplier to these systems, for buyer/supplier transactions. The Program Adapter for HTTP handles incoming XML requests by performing the following actions:

- Recognizing the request and verifying if it is an XML request. If the following three attributes of the request are supported, it can be distinguished it as an XML request.
  - content-type
  - method
  - character encoding

The supported request attributes are specified in the adapter configuration.

- Extracting the input stream of the request.
- Calling the message mapper and passing the content of the input stream.
- Receiving the CommandProperty object representing a WebSphere Commerce command returned by the message mapper.
- Determining the proper device format in which to generate the response.
- Executing the command.
- Sending an XML response message, created by a JSP and based on the viewname specified by the command executed and the device format of the received request.

Each request is treated as a separate session. The credentials of the message are specified in the control area of the message. By default, the Program Adapter for HTTP checks the user ID and password parameters within the message to determine the authenticity of a request. The Program Adapter for HTTP does not support legacy messages because legacy messages do not support the specification of credentials.

The lifecycle of the Program Adapter for HTTP exists throughout the WebSphere Commerce instance. It is initialized when an instance is started unless its configuration parameters are removed or the adapter is not enabled, and it resides as long as the instance is running.

**Note:** The Program Adapter for HTTP is disabled by default.

For architectural information on how WebSphere Commerce handles receiving requests from devices, see the *WebSphere Commerce Programming Guide and Tutorials*. For information on configuring the Program Adapter for HTTP, and other connectivity services, see the WebSphere Commerce Production online help.

### **Listener for WebSphere MQ**

The Listener for WebSphere MQ, is a component of WebSphere Commerce that enables integration with back-end systems by processing inbound messages via WebSphere MQ. The Listener for WebSphere MQ is a combination of using the Adapter for Websphere MQ to retrieve MQ messages, and the Program Adapter, which is called to execute those messages.

The Listener for WebSphere MQ has a set of predefined messages that help integrate WebSphere Commerce business processing with back-end or external system processing. Each incoming message activates processes within WebSphere Commerce to update database tables or perform other operations. Refer to the back-end integration and fulfillment integration message information in the WebSphere Commerce online help for more details on the messages provided. In addition to the existing pre-defined messages, the Listener for WebSphere MQ supports message extensions and new messages.

**WebSphere MQ as middleware:** The Adapter for WebSphereMQ allows you to integrate back-end and external systems with WebSphere Commerce using WebSphere MQ as middleware. The Adpater for WebSphereMQ allows WebSphere Commerce to receive messages from back-end systems and external systems. The supported software is WebSphere MQ Version 5.3 or higher, with the WebSphere MQ client product extension. WebSphere MQ client is the client for WebSphere MQ and is available as a separate download.

You can set up WebSphere MQ as your middleware through the use of bindings or client mode. Bindings mode should be used when WebSphere Commerce is installed on the same machine as the WebSphere MQ server and it connects to the WebSphere MQ server through WebSphere MQ Java using the Java Native Interface (JNI). Since communication is through direct JNI calls to the queue manager API rather than through a network, bindings mode provides better performance than client mode using network connections in general. Client mode should be used when WebSphere Commerce is installed on one machine and the WebSphere MQ server is installed on a back-end system.

To verify WebSphere MQ connections, queues, and channels, run test programs to put messages into and get messages from WebSphere MQ queues. For details, refer to your WebSphere MQ documentation and the *WebSphere Commerce Additional Software Guide*.

### **Adapter for CrossWorlds**

The Adapter for CrossWorlds offers a new mechanism to extend IBM WebSphere Commerce business integration with IBM CrossWorlds. This new adapter uses the CrossWorlds Server Access Interface, an application programming interface (API), that allows an external process to execute a collaboration inside the IBM CrossWorlds InterChange Server (CW ICS). With this adapter, WebSphere Commerce can integrate with external systems by sending synchronous messages to these systems through IBM CrossWorlds. This adapter allows WebSphere Commerce to send a message to the CrossWorlds server and wait for a reply. After getting back the response, the invocation command can continue to proceed with other business logic.

The IBM CrossWorlds system is a suite of software integration products that includes pre-built business logic templates, called collaborations, for common business integration requirements, and various development and management tools. Collaborations define and automate common industry-specific business process steps, such as order management and manufacturing bill of materials management. Collaborations are also used to coordinate and extend the business processes of disparate enterprise software products and to facilitate meaningful data exchange between them.

**Note:** For more information on the Adapter for CrossWorlds, see the *WebSphere Commerce Additional Software Guide* and the WebSphere Commerce Production and Development online help.

### **Integration of WebSphere Commerce with the IBM CrossWorlds InterChange Server:**

WebSphere Commerce is designed to integrate with the IBM CrossWorlds InterChange Server (ICS). This enables WebSphere Commerce to integrate with backend systems through ICS. This integration of WebSphere Commerce with ICS supports three types of message flows:

- The outbound asynchronous message flow allows WebSphere Commerce to send messages to ICS for processing.
- The inbound message flow allows WebSphere Commerce to receive messages from ICS.
- The synchronous message flow allows WebSphere Commerce to send request/reply messages to ICS.

The outbound and inbound flows use the WebSphere Business Integration (WBI) Adapter for WebSphere Commerce. The Request/Reply flow uses the WebSphere Commerce SAI Adapter for ICS.

*WebSphere Commerce outbound process:* Transactions in WebSphere Commerce can trigger messages to be sent to other applications through the WebSphere Commerce messaging system. Messages generated through the messaging system can be used to integrate with other applications through an integration server like the WebSphere Business Integration Server. For example, the OrderCreate XML message can be generated from WebSphere Commerce to create an order in a backend system.

When an order is created in WebSphere Commerce, it can be configured such that the order related information is sent to another system for further processing like fulfillment. Customers using the IBM Websphere Business Integration Server can configure Websphere Commerce and the IBM CrossWorlds ICS such that XML messages from Websphere Commerce can be sent to the IBM CrossWorldsICS for processing. In this case, orders are sent through XML messages from WebSphere Commerce to ICS. This XML message can be processed by IBM CrossWorlds ICS and sent to other systems like SAP.

When you place an order in WebSphere Commerce, the OrderCreate message in XML format is generated and placed in the WebSphere MQ output queue. The WBI Adapter for the WebSphere Commerce agent constantly polls for new messages, which it passes from the WebSphere MQ output queue to the connector controller.

The controller receives the WebSphere Commerce specific business objects and invokes the maps to generate the generic business objects (GBO). The GBOs are passed to the corresponding collaboration, which processes them and sends the request to a backend system. For details on integrating Websphere Commerce and IBM CrossWorlds ICS, see documentation on the WBI Adapter for Websphere Commerce at: <http://www-3.ibm.com/software/websphere/crossworlds/library/doc/v411/welcome.html>.

*WebSphere Commerce inbound process:* Websphere Commerce is designed to integrate with the IBM CorssWorlds ICS to support requests coming from the ICS to Websphere Commerce. The Websphere Commerce messaging system invokes the Websphere Commerce business logic to support request coming from the ICS.

In this scenario, the existing WebSphere Commerce messages are used to invoke business logic triggered requests coming from IBM CrossWorlds ICS.

The WBI Adapter for WebSphere Commerce receives the business objects from collaborations, converts them into XML-format messages using the data handler, and then delivers the messages to the WebSphere MQ queue For details on integrating Websphere Commerce and IBM CrossWorlds ICS, see documentation on the Adapter for CrossWorlds at the following address: <http://www-3.ibm.com/software/websphere/crossworlds/library/doc/v411/welcome.html>.

---

## Configuration tasks

### Configuring messaging services

During administration of the system, Site Administrators can perform the following tasks:

- Add the vehicle (called "transports") for delivering messages
- Configure transports
- View message transport assignments
- Configure message types (add, change, or delete the message description, severity, transport, or device format)
- Add a transport method to a store
- Activate or deactivate a transport method

For more information on configuring the messaging system, see the WebSphere Commerce Production online help.



## Configuring scheduling services

Access the scheduler from the Administration Console. For more information about using the scheduler, refer to the WebSphere Commerce Production online help. For information on maintaining the scheduler, see “Maintaining the Scheduler” on page 135.

## Configuring logging

All logging and tracing configuration is now done in the WebSphere Application Server Administrator’s Console. The following sections cover stand-alone tools only:

- Modifying ECMessage startup options
- Modifying ECTrace startup options

For more information on WebSphere Application Server logging and tracing see the WebSphere Application Server Information Center.

### Enabling tracing components

To enable tracing for your WebSphere Commerce system, you have to use the WebSphere Application Server Administrator’s Console.

#### Enabling tracing at server startup:

1. Open the WebSphere Application Server Administrator’s Console. For information on how to do this, see the WebSphere Application Server Information Center.
2. In the console navigation tree:
  - a. Click **Troubleshooting > Logs and Trace**.
  - b. Click on the server for which you want to enable the trace (for example, WebSphere Commerce Server or Commerce Payments Server).
  - c. Click on **Diagnostic Trace**.
3. Click **Configuration**.
4. Select the **Enable Trace** check box to enable trace, or clear the check box to disable trace.
5. Set the trace specification to the desired state by entering the proper TraceString. See the following tables, Table 6 on page 62, Table 7 on page 63, and Table 8 on page 64 for a list of components you can trace. An example of a proper TraceString is:

```
com.ibm.websphere.commerce.WC_SERVER=all=enabled
```

**Note:** If you are tracing multiple components you should separate them by a colon. For example:  
`com.ibm.websphere.commerce.WC_SERVER=all=enabled:com.ibm.websphere.commerce.WC_ORDER=all=enabled`
6. Select whether to direct trace output to either a file or an in-memory circular buffer.
7. Select **Advanced** from the drop down menu for the desired trace output format for the generated trace.
8. Save the changed configuration.
9. Start the server.

**Note:** For more information on any of these steps see the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

**Enabling tracing on a running server:** You can modify the trace service state that determines which components are being actively traced for a running server by using the following procedure.

1. Open the WebSphere Application Server Administrator's Console. For information on how to do this, see the appropriate section in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).
2. In the console navigation tree:
  - a. Click **Troubleshooting > Logs and Trace**.
  - b. Click on the server for which you want to enable the trace (for example, WebSphere Commerce Server or Commerce Payments Server).
  - c. Click on **Diagnostic Trace**.
3. Select the **Runtime** tab.
4. Change the existing trace state by changing the trace specification to the desired state. See the following tables, Table 6, Table 7 on page 63, and Table 8 on page 64 for a list of components you can trace. An example of a proper TraceString is:

```
com.ibm.websphere.commerce.WC_SERVER=all=enabled
```

**Note:** If you are tracing multiple components you should separate them by a colon. For example:

```
com.ibm.websphere.commerce.WC_SERVER=all=enabled:com.ibm.websphere.commerce.WC_ORDER=all=enabled
```

5. (Optional) Configure the trace output if a change from the existing one is desired.
6. Click **Apply**.

**Note:** For more information on any of these steps see the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

**WebSphere Commerce trace components:** The following are the tracing components in WebSphere Commerce V5.5:

*Table 6. WebSphere Commerce trace components*

| Tracing Component | WebSphere JRAS Extensions Trace Logger       |
|-------------------|----------------------------------------------|
| SERVER            | com.ibm.websphere.commerce.WC_SERVER         |
| CATALOG           | com.ibm.websphere.commerce.WC_CATALOG        |
| DATASOURCE        | com.ibm.websphere.commerce.WC_DATASOURCE     |
| ORDER             | com.ibm.websphere.commerce.WC_ORDER          |
| USER              | com.ibm.websphere.commerce.WC_USER           |
| COMMAND           | com.ibm.websphere.commerce.WC_COMMAND        |
| CF                | com.ibm.websphere.commerce.WC_CF             |
| NEGOTIATION       | com.ibm.websphere.commerce.WC_NEGOTIATION    |
| RAS               | com.ibm.websphere.commerce.WC_RAS            |
| DB                | com.ibm.websphere.commerce.WC_DB             |
| METAPHOR          | com.ibm.websphere.commerce.WC_METAPHOR       |
| SCHEDULER         | com.ibm.websphere.commerce.WC_SCHEDULER      |
| DEVTOOLS          | com.ibm.websphere.commerce.WC_DEVTOOLS       |
| TOOLSFRAMEWORK    | com.ibm.websphere.commerce.WC_TOOLSFRAMEWORK |

Table 6. WebSphere Commerce trace components (continued)

| Tracing Component | WebSphere JRAS Extensions Trace Logger          |
|-------------------|-------------------------------------------------|
| RULESYSTEM        | com.ibm.websphere.commerce.WC_RULESYSTEM        |
| MERCHANDISING     | com.ibm.websphere.commerce.WC_MERCHANDISING     |
| MARKETING         | com.ibm.websphere.commerce.WC_MARKETING         |
| REPORTING         | com.ibm.websphere.commerce.WC_REPORTING         |
| TRANSPORT_ADAPTER | com.ibm.websphere.commerce.WC_TRANSPORT_ADAPTER |
| PERFMONITOR       | com.ibm.websphere.commerce.WC_PERFMONITOR       |
| MESSAGING         | com.ibm.websphere.commerce.WC_MESSAGING         |
| STOREOPERATIONS   | com.ibm.websphere.commerce.WC_STOREOPERATIONS   |
| CACHE             | com.ibm.websphere.commerce.WC_CACHE             |
| EVENT             | com.ibm.websphere.commerce.WC_EVENT             |
| EJB               | com.ibm.websphere.commerce.WC_EJB               |
| CURRENCY          | com.ibm.websphere.commerce.WC_CURRENCY          |
| CATALOGTOOL       | com.ibm.websphere.commerce.WC_CATALOGTOOL       |
| CONTRACT          | com.ibm.websphere.commerce.WC_CONTRACT          |
| UBF               | com.ibm.websphere.commerce.WC_UBF               |
| BI                | com.ibm.websphere.commerce.WC_BI                |
| INVENTORY         | com.ibm.websphere.commerce.WC_INVENTORY         |
| UTF               | com.ibm.websphere.commerce.WC_UTF               |
| RFQ               | com.ibm.websphere.commerce.WC_RFQ               |
| EXCHANGE          | com.ibm.websphere.commerce.WC_EXCHANGE          |
| ACCESSCONTROL     | com.ibm.websphere.commerce.WC_ACCESSCONTROL     |
| AC_UNITTEST       | com.ibm.websphere.commerce.WC_AC_UNITTEST       |
| APPROVAL          | com.ibm.websphere.commerce.WC_APPROVAL          |
| COLLABORATION     | com.ibm.websphere.commerce.WC_COLLABORATION     |
| THREAD            | com.ibm.websphere.commerce.WC_THREAD            |
| SENSITIVE_INFO    | com.ibm.websphere.commerce.WC_SENSITIVE_INFO    |

**JCA connectors trace components:** The following are the JCA connectors tracing components in WebSphere Commerce V5.5:

Table 7. JCA connectors trace components

| Tracing Component    | WebSphere JRAS Extensions Trace Logger |
|----------------------|----------------------------------------|
| JCA E-mail connector | com.ibm.websphere.commerce.jcaemail    |
| JCA File connector   | com.ibm.websphere.commerce.jcafile     |
| JCA JMS Connector    | com.ibm.websphere.commerce.jcajms      |
| JCA Sample Connector | com.ibm.websphere.commerce.jcasample   |

**Commerce Payments trace components:** The following are the Commerce Payments tracing components in WebSphere Commerce V5.5:

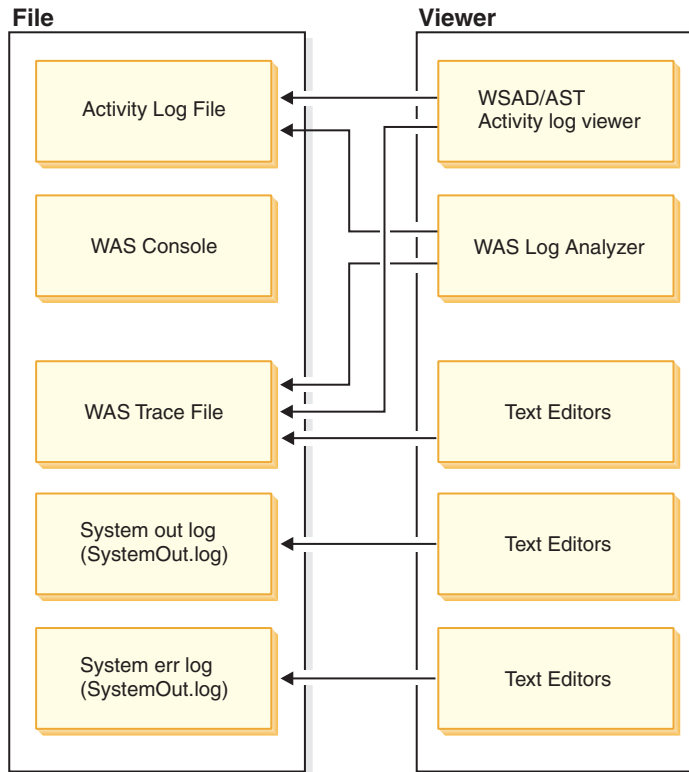
Table 8. Commerce Payments trace components

| Tracing Component                                                                                                                                                                            | WebSphere JRAS Extensions Trace Logger              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| * ( gives you everything)                                                                                                                                                                    | com.ibm.websphere.commerce.payments.*               |
| MPF ( Payments framework)                                                                                                                                                                    | com.ibm.websphere.commerce.payments.MPF             |
| MPFUI ( Payments UI)                                                                                                                                                                         | com.ibm.websphere.commerce.payments.MPFUI           |
| <Cassette_name> ( This enables tracing for any IBM or third party cassette. Cassettes shipped with WebSphere Commerce are: VisaNet, Paymentech, BankServACH, OfflineCard, and CustomOffline. | com.ibm.websphere.commerce.payments.<Cassette_name> |

**Note:** For more information on WebSphere Commerce Payments see the WebSphere Commerce Payments Library.

## Viewing log files

The following diagram depicts the different way log files can be viewed:



### Using Log Analyzer




The Log Analyzer which is part of the WebSphere Application Server, takes one or more service or activity logs, merges all of the data, and displays the entries. Based on its symptom database, the tool analyzes and interprets the event or error conditions in the log entries to help you diagnose problems. Log Analyzer has a special feature enabling it to download the latest symptom database from the IBM Web site. To download the latest updates to the symptom database, use the File -> Update Database -> WebSphere Application Server Symptom Database option for WebSphere Application Server, or WebSphere Application Server Network

Deployment Symptom Database option for WebSphere Application Server Network Deployment in the Log Analyzer interface.

**Service and activity logs:** The WebSphere Application Server server creates the service or activity log file from the activity of the various WebSphere Application Server components. The service or activity log file, `activity.log` is a binary file located at: `installation_directory/logs/activity log`. The Log Analyzer is used to view the service or activity log file, and can merge service or activity log files into one log file. The service or activity log cannot be viewed using a text editor. The Log Analyzer tool is designed for viewing this file. For more information about `activity.log`, see the *WebSphere Application Server System Administration Guide*.

**Viewing the service and activity logs using Log Analyzer:** To view the service or activity log using the Log Analyzer, do the following:

1. Change to the `WASinstall_dir/bin` directory.
2. Run the `waslogbr` script file:

-  `waslogbr.bat`.
-   `waslogbr`.

This starts the Log Analyzer interface.

3. Select **File > Open**.
4. Navigate to the directory containing the service or activity log file.
5. Select the service or activity log file and click **Open**.
6. To analyze the records, right click on a record in the tree on the left, select **UnitOfWorkView** from the right-click menu, and select **Analyze**.


Now any records with a green check mark next to them match a record in the symptom database. When you select a check-marked record, you will see an explanation of the problem in the lower-right-hand pane. For more information on the Log Analyzer user interface, see WebSphere Application Server Information Center.

## Using a text file

To view your log files, you can use any text editor. Your log files can be found in the `logs` directory where the WebSphere Commerce server is installed. Normally this is in the `../appserver/logs/` directory but can be changed. The following is an example of the path you can use to find the log files.

`WAS_installdir/logs/WC_instanceName/trace.log`

**Note:**  Use a text editor other than `vi` to view the generated WebSphere Commerce log files. The lines are too lengthy for `vi` and as a result, cannot be viewed properly.

## Configuring LDAP

Once you have installed WebSphere Commerce, you may choose to use the LDAP server as the member repository, or you may start with using the database as the member repository and switch to using the LDAP server later. For more information on configuring LDAP see, "Configuring WebSphere Commerce for LDAP", in the *WebSphere Commerce Additional Software Guide*.

## Configuring rules services

To configure rules services, log on to the Administration Console and from the **Rules Services** menu, click **Administration**. Site Administrators can perform the following rules services tasks using the Administration Console:

- Add a rule service
- Edit a rule service
- Check the status of a rule service
- Refresh a rule service
- Delete a rule service
- Start a rule service
- Stop a rule service

For more information on the tasks to configure rules services see the WebSphere Commerce online help.

## Configuring the staging server

### Using staging server utilities

Staging utilities exist on the staging server and connect to both the production and staging databases using standard DB2/JDBC communication. Administrators must make sure that the DB2 client/server environment is setup properly. The DB2 client talks to the DB2 server through a TCP/IP port. The number is set in the DB2 instance variable SVCENAME. You must be aware of this port number and make sure the proxy will not block this port. By default, DB2 uses 50000. For more information on setting up JDBC and the DB2 client/server environment, refer to the *DB2 Administration Guide*

Once your environment is set up and you are ready to copy data from the staging server to the production server, push the data using the Stage Propagate utility. Tables that can be copied are listed in staging tables. Any new tables that you have created to store custom data can also be copied to the staging server after configuring the staging server for customized tables.

The STAGLOG table acts as an internal log. Whenever you change a record in a table on the staging server, a trigger records this change in the STAGLOG table. For each modified record, a trigger records the type of modification (insert, delete, or update), the name of the table where the record resides, and the record's primary key or unique index. After changing and testing database records on the staging server, propagate the changes back to the production server using the Stage Propagate utility.

The database tables covered by the staging server should never be updated on the production database during a staging session. A staging session begins when you use the Stage Copy utility to copy the production database to the staging database. A staging session ends when you use the Stage Copy utility to begin another staging session. After your data is copied from the production database to the staging database, the staging database and the production database are synchronized in terms of the tables covered by the staging server. Once the tables are synchronized, changes to the tables on the production database are not permitted. You can only update the staging database, and then use the Stage Propagate utility to propagate the changes to the production database. If you update both databases, your propagation will likely fail due to a potential key conflict or a reference integrity violation. If you must update the production

database during a staging session, use the Stage Copy utility to synchronize your databases and begin a new staging session.

To guarantee that the tables are never updated on the production database during a staging session, the tables must be under the control of a Site Administrator only. In some cases, the staging tables in your production database are updated by an individual customer or merchant after your the data has been copied using staging copy. For example, you cannot prohibit a merchant from modifying the OFFER table in the production database after copying to the production database.. In this situation, you cannot use the staging server. However, RFQ objects are an exception. When you create RFQ objects on the production database, rows are inserted into trading tables in the production database. If you are creating contracts in the staging database, you are also inserting rows into the trading tables in the staging database. In this case, you are updating the same tables on both the staging database and the production database.

Note that there are some limitations for the staging server when using RFQ objects. You also need to run the Stage Check utility to find potential unique index key conflicts and correct them before running the Stage Propagate utility to propagate the changes to production database.

In a typical business-to-consumer site, tables can be divided into two groups: configuration data and operation data. The configuration tables contain data such as stores, catalogs, catalog entries, languages, taxes, and discounts. These tables are under Site Administrator control; an individual customer cannot modify the tables. The operation tables contain data such as customer information, address, orders, and SET-related data. Customers can modify operation tables. The staging server only covers configuration tables. Refer to WebSphere Commerce staging tables for a list of tables covered by the staging server.

It is also important to ensure that tables covered by the staging server do not contain any foreign key references to operation tables. Otherwise, the propagation could fail due to a potential primary key deletion from the production database. Before you use the staging server, you should ensure that only the organization owns the operational data, and not the individual user, such as a catalog administrator.

You should be aware of the following before using the staging server:

- Any new image files, HTML files, or JSP files that are referenced by the staged records must be manually copied from the staging server to the production server.
- The staging server cannot copy and propagate database schema changes, image files, HTML files or JSP files. For example, if you create a new index or table in a staging database, you must manually create the index or table in production database.
- The Stage Propagate utility cannot propagate records loaded by the Loader package (load mode) or the DB2 Load utility since both bypass the staging triggers. If you have used either utility, use the Stage Copy to resynchronize your database tables and begin a new staging session. You should never use the Loader package (load mode) or DB2 Load on a staging database or a production database during a staging session.
- After using the Stage Copy utility, you must stop and restart the staging server.
- The staging server does not support DB2 Text Extender.

- Do not run the Database Cleanup utility on the staging server except to clean the STAGLOG table.
- There are certain staging server limitations. Ensure that you understand the staging server limitations before using the staging server. See “Staging server limitations” on page 34 for more information.

**Stage Copy utility:** The Stage Copy utility, `stagingCopy.sh` for iSeries, copies data from the production database to the staging database. You can copy data in site-related tables, merchant-related tables, or individual tables. You can also clean the staging database before the Stage Copy utility by using the `cleanup_stage_db` parameter in the command syntax. If you specify `yes`, the Stage Copy utility cleans all staging tables before copying the data. This may impact other tables with the delete cascade. If you specify `no`, the Stage Copy utility will not delete anything from the staging tables. Your copy might fail if your copy data generates a conflict or duplicates a key on the primary key or unique indexes. To use the Stage Copy utility to only clean your staging database, specify `only`.

The Stage Copy utility and the Stage Propagate utility divide database data into two scope levels: site-related and merchant-related. The site scope includes data common to all merchants in the system. For example, the language and country or region code used by the system. The merchant scope includes data related to individual merchants. For example, store information is customized for individual merchants, and rows from the store tables could be specific for each merchant. Certain database tables contain both site and merchant information. If you specify the scope parameter to `_all_` during the Stage Copy utility, the site data will be copied, followed by all merchant data. If you specify the scope to `_site_`, only the site data will be copied. If you specify the scope to `_merchant_`, only the merchant data will be copied. Note that you cannot copy data for an individual merchant, only all merchants. If you do not set the scope to `_all_`, copy the site data before the merchant data since the site data is used by all merchants. Otherwise, your copy will fail due to a mismatch between the foreign key and the primary key. When you use the `cleanup_stage_db` to clean the site data, note that the merchant data can be deleted because of the delete cascade. You should clean the merchant data followed by the site data, and then copy the site data followed by the merchant data if you do not set the scope to `_all_`.

Another option with the Stage Copy utility is `script_file` parameter. By specifying a script file name, the Stage Copy utility generates an SQL script file which uses export and import to copy the production database to the staging database based on the specified scope. Delete statements are also generated to clean the staging database if you use the default value or specify `cleanup_stage_db` to `yes`. The script file is located in the directory where you start your Stage Copy utility. The script file speeds your database copy process using export and import. You can also modify the behavior of the Stage Copy utility by changing the generated script file. For example, you can change the script file to use the DB2 load utility instead of the import utility, which will further speed up the copy process. Note that the generated script exports all tables to the directory where you run the utility. Ensure that you have enough disk space.

It is important to understand the transaction scope. When cleaning the staging database, the Stage Copy utility commits the transaction after cleaning each table. When copying the data, the Stage Copy utility commits the transaction after copying each table and synchronizing the KEYS table. For generated scripts, the transaction scope is slightly different because of the DB2 import utility. The DB2 import utility automatically commits the transaction after finishing the import. The



transaction is committed before synchronizing the KEYS table. Consequently, synchronizing the KEYS table is done in a separate transaction.

You can specify a table to clean or copy using the `dbtable` parameter. Note that when you specify which table to clean or copy, the table may not be isolated. Certain tables are related to each other by referential constraints. If you clean a specified table, you will also clean the child tables by the delete cascade. If you copy a specified table, you should first copy the parent table. Otherwise, your clean or copy will fail.

The Stage Copy utility is configurable and extensible. To handle your customized tables, there are some conditions your tables must meet and you must set up in the staging configuration tables. Before you can use the Stage Copy utility, you must follow the steps in configuring the database. The Stage Copy utility deletes all records from STAGLOG table if the command is successful.

**Note:** You cannot use the Stage Copy command if RFQs are on your system. For more information, refer to “Staging server limitations” on page 34.

**Stage Check utility:** When the configuration and operation data share the same table, a unique index key conflict might occur between the staging database and the production database. Before propagating your changes to the production database, use the Stage Check command, or `stagingCheck.sh` for iSeries, to determine any potential unique index conflicts and correct the conflicts before propagation.

When you are using RFQs on your production database and creating contracts on your staging database, you are updating the same tables on both databases. For example, a Site Administrator creates a contract on a staging database, which will insert one row in the TRADING table (and other tables) on the staging database. At the same time, a user creates an RFQ on the production database, which will insert one row in TRADING table (and other tables) on the production database. The two rows might have the same unique index value in the TRADING table. When propagating the contract from the staging database to the production database, a unique index key conflict is generated, and the propagation will fail. Before propagating, use the Stage Check utility to find the conflicted unique index key and correct them. After that, you can propagate the changes.

When you use the Stage Check utility, specify the `-scope` parameter as `_unique_index_` to check the potential key conflicts for the delta changes in the staging database. For all insert and update operations, it will check the potential index key conflict for all tables specified in the STGUINDTAB table. For each table, it will go over all unique indexes, and check if there is potential key conflict between production database and staging database. If there are potential key conflicts, it will report the table name, unique index, and conflicted key value.

The Stage Check command does not change your database; it reports the potential key conflict, which must be resolved. When using this command, specify the `-sourcedb` parameter as the staging database. The Stage Check command does not function properly if you specify the production database as your source database.

The staging check utility is configurable and extensible. You can add more tables or your customized tables into STGUINDTAB table and run the Stage Check command to verify any potential key conflicts.

**Note:** Always ensure that your configuration and operation data do not share the same table.

**Stage Propagate utility:** After changing and testing the database records on the staging server, check for potential unique index key conflicts and correct them using the Stage Check utility. You are now ready to propagate the changes to the production database.

The Stage Propagate utility, or stagingProp.sh for iSeries, moves changes from the staging database to the production database. The Stage Propagate utility uses the STAGLOG table to identify the changed records in the staging database, then updates these records in the production database. Processed records are indicated in the STAGLOG table by a 1 in the STGPROCESSED column.

You can specify the scope parameter to select the type of data for propagation. Set to `_site_`, all changed site data is propagated from the staging database to the production database. Set to `_merchant_`, the changed data of all merchants is propagated. You cannot propagate individual merchant data. Set to `_all_`, both site and merchant data is propagated.

Using the `dbtable` parameter, you can propagate a specific table. Ensure that the parent table has been propagated before you specify a table.

The transaction scope for the Stage Propagate utility is different than the Stage Copy utility. Each run of the Stage Propagate utility command counts as one transaction. For example, if you specify the `scope` as `_site_`, the Stage Propagate utility will begin a new transaction for all modified site data and commit the transaction after a successful propagation. If the propagation fails, the propagation rolls back and the state of your production database is the same as before.


The Stage Propagate utility is configurable and extensible. Before propagating your customized tables, the tables must meet certain conditions. For details, refer to configuring the staging server for customized tables. Before you can use the Stage Propagate utility, you must follow the steps in configuring the database.

## Configuring the database






Before using the Stage Copy utility, the Stage Propagate utility, or the Database Cleanup utility, you may wish to do the following:

### Notes:

1. The following steps are just a suggestion, and should only be used if there are problems encountered with the Stage Copy utility.

2. The following does not apply to iSeries unless indicated by 






1. Set the PATH environment variables.

2.      If you are using a DB2 database, configure the staging database and the production database by issuing the following commands:

```
db2 update db config for db_name using LOGPRIMARY 80
db2 update db config for db_name using LOGBUFSZ 512
db2 update db config for db_name using DBHEAP 2048
db2 update db config for db_name using APPLHEAPSZ 2048
db2 update db config for db_name using PCKCACHESZ 8200
```

where `db_name` is the name of your database.

**Notes:**

- a. The default STMTHEAP size is 60000.
  - b. The default LOCKLIST is 2400.
  - c. The default STAT\_HEAP\_SZ is 2048.
  - d. The default APP\_CTL\_HEAP\_SZ is 4096.
3.     Increase the buffer pool size for improved performance. Determine the optimum buffer pool size based on your DB2 database size and available memory. Run the following command to change the default buffer pool size:
- ```
db2 connect to db_name
db2 alter bufferpool IBMDEFAULTBP size n
db2 terminate
```
- where *n* is the optimum buffer pool size.
4.  Log on as a user profile with secofr authority and a cssid of something other than 65535.

Configuring the staging server for customized tables

To use the staging server with your customized database tables, do the following configuration:

1. Identify your customized table scope (site data, merchant data, or site and merchant data).
2. Create the trigger for your database table using the corresponding trigger examples based on your table's scope and index type.
3. Insert the customized tables into the STGSITETAB, STGMERTAB, and STGMRSTTAB tables.
 - For site tables, insert only into STGSITETAB.
 - For merchant tables, insert only into STGMERTAB.
 - For tables which contain both site and merchant data, insert into STGSITETAB, STGMERTAB, and STGMRSTTAB.

Note: You must ensure that all the parent tables have been inserted properly and that the TABNBR column of the parent tables is less than that of the child table. If your customized table is the parent table of a WebSphere Commerce table, you also need to ensure that your table's TABNBR column is less than that of the child tables.

Configuring connectivity

For information on configuring the various adapters, see the *WebSphere Commerce Additional Software Guide*, and the WebSphere Commerce Production online help.

Configuring WebSphere Commerce Payments

Before you can configure Payments, do the following:

- Ensure that WebSphere Commerce Payments is part of your WebSphere Commerce installation.
- Create a WebSphere Commerce Payments instance.
- Use the Configuration Manager to add a cassette to the WebSphere Commerce Payments instance.
- Start the WebSphere Commerce Payments instance, and the WebSphere Commerce instance.
- Create a merchant and Merchant Administrator for that merchant.

To configure a cassette and use Payments, you must log on to WebSphere Commerce Payments as a Merchant Administrator. WebSphere Commerce Payments is installed with the CustomOffline Cassette and OfflineCard Cassette. The minimum framework that these cassettes support is WebSphere Commerce Payments Version 5.5. For more information, see the *WebSphere Commerce Installation Guide*.

Following are the WebSphere Commerce and Payments default installation directories:

Table 9. Default installation directories

Default installation directory	Description
<i>WC_installdir</i>	Windows: <i>drive:</i> \WebSphere\CommerceServer55 AIX: <i>/usr/WebSphere/CommerceServer55</i> Solaris, Linux: <i>/opt/WebSphere/CommerceServer55</i> 400: <i>/QIBM/ProdData/CommerceServer55</i>
<i>Payments_installdir</i>	Windows: <i>drive:</i> \WebSphere\CommerceServer55\payments AIX: <i>/usr/lpp/WebSphere/CommerceServer55/payments</i> Solaris, Linux: <i>/opt/WebSphere/CommerceServer55/payments</i> 400: <i>/QIBM/ProdData/CommercePayments/55</i>

For detailed information on configuring WebSphere Commerce Payments, see Appendix C, “WebSphere Commerce Payments Tutorial,” on page 155.

Chapter 4. Hosted store administration

There are several tasks that a Site Administrator has to perform in a hosted store scenario. This chapter outlines the tasks that a Site Administrator performs in the following categories:

- Administering resellers
- Administering distributors
- Site administration

Administering resellers

In a hosted store scenario, the Site Administrator is responsible for setting up the reseller organization and maintaining a relationship with the reseller. This coordination involves the following tasks:

- Sending e-mail approvals of new reseller organization registration.

Once a reseller has registered their organization, it is the Site Administrator's responsibility to send an e-mail to the reseller notifying them that their request for an organization has been approved. The approval of actual organization registration is performed in the Organization Administration Console. For more information on the Organization Administration Console and the tasks that can be performed within it, see the WebSphere Commerce Production online help.

Once the reseller organization has been approved, resellers in the Demand chain business model can modify their stores using the WebSphere Commerce Accelerator. Each store is created in closed state however, the reseller can use the WebSphere Commerce Accelerator to make updates and changes to the store, and eventually open it. In the Hosting business model, each store is created in a suspended state and needs to be resumed by the Site Administrator or the Channel Manager. The merchant, in this case, cannot make any changes to the store until it has been resumed. For more information on the business models, see *WebSphere Commerce Fundamentals*, and the *WebSphere Commerce Store Development Guide*. For more information on the WebSphere Commerce Accelerator, see the WebSphere Commerce Production online help.

- Sending the merchant hosted store approval. (only for stores using the Hosting business model)

Once the hosted store has been set up, the Site Administrator sends the merchant an e-mail with the hosted store request approval. This e-mail must also contain a link to the new hosted store, and instructions on how to administer it.

Note: For more information on administering a reseller organization, see "Administering the reseller organization", in the WebSphere Commerce Production online help.

Administering distributors

In a hosted store scenario, the Site Administrator is responsible for setting up distributors and creating a distributor relationship with the reseller. This administration involves the following tasks:

- Creating a new distributor.

The Site Administrator can create a new distributor by creating a distributor service agreement. Information to be added to the contract can be obtained by contacting IBM directly. For more information on the distributor service agreement, see the WebSphere Commerce Production online help.

Note: This task can also be performed by the Channel Manager.

- Changing distributor settings.

At times the Site Administrator may receive requests to change some of the distributor settings such as the time-out setting on connection to a distributor. Distributor settings can be changed by accessing the database through SQL statements. For more information, see the *WebSphere Commerce Store Development Guide*.

- Configure and install the Adapter for CrossWorlds

The Adapter for CrossWorlds is the connector used to establish interactions between WebSphere Commerce and external and backend systems for the hosted store. The Site Administrator is responsible to configure and install the adapter to ensure connectivity between reseller and distributor sites. Interactions between WebSphere Commerce with the SAP R/3 backend system using WebSphere MQ can also be created, and must be set up by the Site Administrator. For more information on IBM CrossWorlds and WebSphere MQ, refer to the *WebSphere Commerce Additional Software Guide*.

Site administration

The site administration tasks specific to hosted stores include the following:

- Operating the staging server. For more information see, “Staging server” on page 33.
- Operating DBClean. For more information see “Database Cleanup utility” on page 95.
- Operating WebSphere Commerce Analyzer. For more information, see the WebSphere Commerce installation CD.
- Starting and stopping the WebSphere Commerce server. For more information see “Starting and stopping WebSphere Commerce” on page 11.

Chapter 5. Dynamic caching

This section describes the dynamic caching of WebSphere Commerce servlet or JavaServer Pages (JSP) results.

WebSphere Commerce servlet or JSP result caching

When a customer clicks a link to view a product or category page, most of the time is spent parsing the HTTP request, accessing the database, and dynamically creating the page. Heavy site traffic and many product and category entries in the database can further increase the time it takes for servlets or JavaServer Pages (JSP files) to load.

Most HTTP requests on the server will be for catalog information. The WebSphere Commerce commands — `CategoryDisplay`, `ProductDisplay`, `TopCategoriesDisplay`, and `StoreCatalogDisplay` — retrieve information from your database, and display the result as a JSP page. If the catalog information has not changed since it was last viewed, the servlet or JSP file does not need to be re-executed the next time a customer requests it. Serving an equivalent static servlet or JSP file stored in a cache would be faster. Caching of the servlet or JSP can be constructed by defining cache-entry elements in the `cachespec.xml` file located in the `WEB-INF` directory of the Web module.

If the cache entry corresponding to the page being accessed is not in memory, it will be generated dynamically. The page is then stored to memory, and will not have to be regenerated until the data it is based on is modified.

Note: WebSphere Commerce dynamic caching and URL rewriting cannot interoperate. With URL rewriting turned on, you need to disable WebSphere Commerce dynamic caching and you should not cache your servlet or JSP files. For more information URL rewriting, see the chapter on session management in the *WebSphere Commerce Security Guide*.

WebSphere Application Server dynamic cache

WebSphere Commerce utilizes the WebSphere Application Server dynamic cache service for caching servlets or JSP files and commands that extend from the WebSphere Application Server `CacheableCommand` interface. The dynamic cache service, servlet caching and disk offload are enabled by default, during the creation of a WebSphere Commerce instance.

For more information on the WebSphere Application Server dynamic cache, refer to the topic “Improving performance through the dynamic cache service” in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Dynamic cache

Caching the servlet or JSP file results improves application performance. WebSphere Application Server consolidates several caching activities, including servlets, Web services, and WebSphere commands into one service called the dynamic cache. These caching activities work together to improve application

performance, and share many configuration parameters, which are set in an application server's dynamic cache service.

You can use the dynamic cache to improve the performance of servlet and JSP files by serving requests from an in-memory cache. Cache entries contain servlet output, results of servlet execution, and metadata.

For more information, see the topic "Improving performance through the dynamic cache service" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Enabling the dynamic cache service and servlet caching

To enable caching, you should enable the dynamic cache service and configure servlet caching. For information to perform these step, see the topics "Enabling globally the dynamic cache service" and "Configuring servlet caching" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Enabling the Dynamic Cache Monitor

WebSphere Application Server provides a Dynamic Cache Monitor application for displaying cache statistics, Edge Side and disk statistics, cache entries, dependency IDs, and cache policy information. In order to inspect the contents and behavior of the WebSphere Application Server dynamic cache, you should install the WebSphere Application Server Dynamic Cache Monitor. To use the Dynamic Cache Monitor, install the CacheMonitor.ear file, located in the `installableApps` subdirectory under the `WAS_install_dir` directory, on each application server that uses dynamic caching.

For WebSphere Commerce, it is recommended that you use the virtual host `VH_instance_name_admin` (for example, `VH_demo_admin`) instead of `VH_instance_name` for the Dynamic Cache Monitor, for security reasons. You can add or change virtual host names from the WebSphere Application Server Administrative Console.

You can access the Web application using a Web browser with the following Web address:

```
http://host_name:port/cachemonitor
```

However, for a more secure access, it is recommended that you access the Administration host machine:

```
https://admin_host_name:port/cachemonitor
```

For example, if the virtual host `VH_instance_name_admin` is used for installing the Cache Monitor, then the Cache Monitor can be accessed as:

```
https://admin_host_name:8002/cachemonitor
```

Notes:

1. You need to launch one dynamic cache monitor for every WebSphere Commerce instance.
2. Whenever you want to use the Cache Monitor and you have WebSphere Application Server EJB security enabled, you will need to perform some additional set up as documented in the section "Security configuration for the Dynamic Cache Monitor" in the *WebSphere Commerce Security Guide*.

For more information on installing the Dynamic Cache Monitor see the topic "Displaying cache information " in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Tree View for Dependency IDs for WebSphere Dynamic Cache Monitor

WebSphere Commerce provides a Tree View tool, an extension of the Cache Monitor, which provides a tree view of the dependency ID in memory. The Tree View tool provides a tree view of the dependency ID. You can expand or collapse any tree node to get a hierarchical view, as well as invalidate a cache entries group under that node. The original view will be untouched instead a new link will appear as "Dependency IDs Tree View".

For information on installing and using the Tree View tool, see the README file for the "Tree View for Dependency IDs for WebSphere Dynamic Cache Monitor" located in the following directory:

 `WC_installdir/features/dynacache/CacheMonitor/`

 `WC_installdir\features\dynacache\CacheMonitor\`

Configuring cacheable objects

Cacheable objects are defined inside the `cachespec.xml` file, found inside the Web application archive (WAR) WEB-INF or enterprise bean WEB-INF directory. You can place a global `cachespec.xml` file in the application server properties directory, but the recommended method is to place the cache configuration file with the deployment module. The root element of the `cachespec.xml` file is `<cache>`, which contains `<cache-entry>` elements.

To specify the cache entry for servlet or JSP result caching, add the following section to the `cachespec.xml` file:

```
<cache-entry>
  <class>servlet</class>
  <name>name</name>
  ...
</cache-entry>
```

where *name* is the relative Web path or servlet mapping of the servlet or JSP

To specify the cache entry for command caching, add the following section to the `cachespec.xml` file:

```
<cache-entry>
  <class>command</class>
  <name>name</name>
  ...
</cache-entry>
```

where *name* is the complete path to the command class, for example, `com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmdImpl`.

Within a `<cache-entry>...</cache-entry>` element, you can develop cache-ID, dependency-ID, and invalidation rules. To cache an object, WebSphere Application Server has to be able to generate unique IDs for different invocations of that object. The `<cache-id>` element performs this task. Each cache entry can have multiple

cache-ID rules that execute in order until either a rule returns a non-empty cache-ID or no more rules remain to execute. If none of the cache-ID generation rules produce a valid cache ID, then the object is not cached.

Note: For this release, when specifying the cache entry for command caching, only command invalidation using the `<invalidate>...</invalidate>` is supported. (You cannot use the `<cache-id>...</cache-id>` or `<dependency-id>...</dependency-id>` elements for command caching.)

```
<cache-entry>
  <class>command</class>
  <name>name</name>
  <invalidate>
    ...
  </invalidate>
</cache-entry>
```

The dynamic cache responds to changes in the `cachespec.xml` file. When new versions of `cachespec.xml` are detected, the old policies are replaced. Objects cached through the old policy file are not automatically invalidated from the cache. They are either reused with the new policy or eliminated from the cache through its replacement algorithm. For more information about the `cachespec.xml` file, see the topic “Cachespec.xml file” in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>). In addition, see the *WebSphere Commerce Store Development Guide* for additional considerations when developing your store.

Note: For requests using the PVC adapter (for a pervasive computing device), you can only use the URL parameters as the cache keys.

Caching defaults

During installation, WebSphere Commerce sets up the caching system with the following defaults:

- Dynamic caching is enabled.
- Disk offload is enabled.
- Servlet caching is enabled.
- Default cache policies (supplied as samples) for each business model caches the JSP files for the following display commands :
 - CategoryDisplay
 - ProductDisplay
 - StoreCatalogDisplay
 - TopCategoriesDisplay

You need to manually merge the cache entry from:

 **AIX** **400** **Linux** **Solaris**

`WC55_installdir/samples/dynacache/business_model/cachespec.xml`

to:

`WAS_installdir/installedApps/cell_name/WC_instance_name.ear/Stores.war/WEB-INF/cachespec.xml`

 **Windows**

`WC55_installdir\samples\dynacache\business_model\cachespec.xml`

to:

`WAS_installdir\installedApps\cell_name\WC_instance_name.ear\Stores.war\WEB-INF\cachespec.xml`

Refer to the *WebSphere Commerce Sample Store Guide* or *WebSphere Commerce Store Development Guide* for details on enabling the cache policies in a store.

Full page and fragment caching

The method by which the WebSphere Application Server dynamic cache caches JSP files is based on how the JSP is written. If the page output for a particular WebSphere Commerce command always produces the same result based on the URL parameters and request attributes, then this page output can be cached with the cache-entry using the property element, consume-subfragments (CSF) along with the WebSphere Commerce controller servlet (`com.ibm.commerce.server.RequestServlet`) as the servlet name. When the cache-entry is defined in this manner, then the page output is cached in a manner known as *full page* caching. The big advantage of using consume-subfragments with the controller servlet is performance, but if this mechanism is used to cache the web pages, then the page output can not have personalized information on it.

If the page output has sections that is user-dependent, then the page output is cached in a manner known as *fragment* caching. That is, the JSP pages are cached as separate cache entries, and get reassembled when they are requested. For fragment (JSP) caching, WebSphere Commerce has to execute the command to determine which JSP is to be executed before the dynamic caching mechanism can determine if the JSP can be served up from cache or not. The advantage of this method is flexibility, because different cache entries can be reassembled together to form a page, based on user information.

Full page caching

When the property element consume-subfragments (CSF) is used, the parent entry (the one marked with CSF) will include all the content from all fragments in its cache entry, resulting in one big cache entry that has no includes or forwards, but rather, the content from the whole tree of entries.

When a servlet is cached, only the content of that servlet is stored. The cache includes place holders for any other fragments to which it includes or forwards. Consume-subfragments (CSF) tells the cache not to stop saving content when it includes a child servlet. The parent entry (the one marked CSF) will include all the content from all fragments in its cache entry, resulting in one big cache entry that has no includes or forwards, but the content from the whole tree of entries. This can save a significant amount of application server processing, but is typically only useful when the external HTTP request contains all the information needed to determine the entire tree of included fragments.

For example, if the `<cache-entry>` is defined as follows:

```
<cache-entry>
  <class>servlet</class>
  <name>com.ibm.commerce.server.RequestServlet.class</name>
  <property name="consume-subfragments">true</property>
  <property name="save-attributes">>false</property>
  <property name="store-cookies">>false</property>

  <!-- StoreCatalogDisplay?storeId=<storeId> -->
  <cache-id>
    <component id="" type="pathinfo">
      <required>true</required>
      <value>/StoreCatalogDisplay</value>
    </component>
    <component id="storeId" type="parameter">
```

```

    <required>true</required>
  </component>
</cache-id>
</cache-entry>

```

Note that when the `save-attributes` property is set to `false`, the request attributes are not saved with the cache entry. When the `store-cookies` property is set to `false`, the request cookies are not saved with the cache entry.

In the above example, the cache servlet entry will contain a consumed include of `StoreCatalogDisplay.jsp` which is the JSP file forwarded by the `StoreCatalogDisplay` command.

Fragment cache

Each dynamically included JSP file has to have its own `<cache-entry>` defined in the `cachespec.xml` file in order to be served up by the dynamic cache when it receives a request. Otherwise, each dynamically included JSP file will be re-executed for each request. For example, consider if `StoreCatalogDisplay.jsp` dynamically includes `header.jsp` and `footer.jsp` and you only set up a `<cache-entry>` for the `StoreCatalogDisplay.jsp`. Then, when you request the `StoreCatalogDisplay` page, the `header.jsp` and `footer.jsp` files get executed if they are not cached. Here is an example of how to define the `<cache-entry>` for `StoreCatalogDisplay.jsp`:

```

<cache-entry>
  <class>servlet</class>
  <name>/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp</name>
  <property name="save-attributes">false</property>

  <cache-id>
    <component id="storeId" type="parameter">
      <required>true</required>
    </component>
    <component id="catalogId" type="parameter">
      <required>false</required>
    </component>
  </cache-id>
</cache-entry>

```

Cache servlet filtering

Prior to WebSphere Commerce Version 5.5, WebSphere Commerce provided its own caching mechanism. Using the previous mechanism, Web pages could be cached based on two types of methods:

- Session independent (SI) caching - Pages were cached based on URL parameters
- Session dependent (SD) caching - Pages were cached based on URL parameters, user's language, preferred currency, parent organization, contract IDs, and member groups

Cache IDs for SI caching were generated based on the URL parameters, and for SD caching, the cache IDs were created with the URL parameters plus the session information.

In order to provide the same functionality as the previous session dependent caching but using the WebSphere Application Server dynamic caching mechanism, WebSphere Commerce 5.5 introduces a servlet filter known as the *cache filter*. This cache filter is designed to setup the request attributes from the session information to be used by the dynamic cache to construct the cache ID. Since the session

information is set by the WebSphere Commerce server runtime, the cache filter will not be able to set all of the request attributes until the second request against the Web site.

The following table lists the request attributes that are set by the cache filter:

Table 10. Request attributes that are set by the cache filter

Request attributes	Description
DC_curr	User's preferred currency
DC_lang	User's preferred language
DC_porg	User's parent organization
DC_cont	User's current contract
DC_mg	User's explicit member groups
DC_storeId	Store identifier
DC_userId	User's identifier

Since a user can be eligible for multiple contracts and can belong to multiple member groups, the request attributes DC_cont and DC_mg might contain multiple values. For such a user, the values are sorted and concatenated together with a semicolon(";") as a separator. In addition, multiple contract and member group request attributes will be defined. (for example, DC_cont0, DC_cont1, ... DC_contN where N is the number of contracts to which the user is entitled). For example, if a user is eligible for contracts 10004 and 10005, then the following request attributes will be set up: DC_cont is 10004;10005, DC_cont0 is 10004, DC_cont1 is 10005.

The purpose of setting request attribute DC_cont is to allow construction of a cache ID that has a limited number of components and the purpose of setting individual request attributes DC_cont0, DC_cont1, ..., DC_contN is to allow construction of dependency IDs for more granular cache invalidations.

Since the member group information is not part of the session data, the cache filter has to retrieve this information from the database based on the user ID. In order to prevent performance degradation due to repeating database queries, the cache filter uses WebSphere Command Caching to accomplish this task. A new command - com.ibm.commerce.dynacache.command.MemberGroupsCacheCmdImpl - that extends directly from the WebSphere Command Framework is used to cache the member groups to which users belong, based on user IDs. WebSphere Commerce adds the following cache entry to the cachespec.xml to notify the dynamic caching service to cache this command:

```
<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.dynacache.commands.MemberGroupsCacheCmdImpl</name>
  <cache-id>
    <component type="method" id="getUserId">
      <required>true</required>
    </component>
  </cache-id>
  <dependency-id>DC_userId</dependency-id>
  <dependency-id>DC_userId
    <component type="method" id="getUserId">
```

```

        <required>true</required>
    </component>
</dependency-id>
</cache-entry>

```

WebSphere Application Server dynamic caching will cache the command object of class `MemberGroupsCacheCmdImpl` based on its `get` method `getUserId()` as its cache ID. Therefore, next time there is a request for member groups of the same user ID, dynamic caching will return the cached command object with an already computed member group. This member group information is then readily accessible using the command method `getMemberGroups()` which the cache filter calls to set the member groups for this user in the request attribute, `DC_mg`.

Note: WebSphere Commerce uses WebSphere Command Caching internally such as with `MemberGroupsCacheCmdImpl` in the cache filter above; however, WebSphere Commerce does not officially support WebSphere Command Caching of WebSphere Commerce commands.

Store page caching strategy

For information on caching considerations when designing your store pages see the section “Caching your store pages” in the *WebSphere Commerce Store Development Guide*. This includes guidance on the following issues:

- What pages should be cached?
- Should whole pages or page fragments be cached?
- Developing a caching strategy

In addition, see the section discussing implications of store relationships on caching in the *WebSphere Commerce Store Development Guide*.

Cache invalidation mechanisms in WebSphere Commerce

Cache entries generated by the WebSphere Application Server dynamic cache service contains information including the servlet output, the results of servlet execution (such as calls to other servlets or JSP files), and metadata about the entries such as the timeout and entry priority information. From time-to-time, WebSphere Commerce needs to verify that this information is current, consistent, and accurate while it remains in the cache. WebSphere Commerce therefore requires efficient mechanisms for identifying and removing cache information that is no longer valid. The WebSphere Application Server dynamic cache service provides support to maintain the cache information with a process called *cache invalidation*. Cache invalidation allows WebSphere Commerce to exploit the different invalidation mechanisms provided by the service to perform the invalidation under different circumstances. These mechanisms vary from rule-based, time-based, group-based, to programmatic approaches.

This section only focusses on describing how to perform cache invalidation in WebSphere Commerce and is not intended to cover the general caching aspects. It describes the invalidation mechanisms provided by the WebSphere Application Server dynamic cache:

- Defining the invalidation policies in `cachespec.xml`
- Removing cache entries through the Cache Monitor provided by WebSphere Application Server
- Invoking the WebSphere Application Server dynamic cache invalidation APIs programmatically inside the WebSphere Commerce application

Defining the invalidation policies in cachespec.xml

WebSphere Application Server dynamic caching supports the configuration of cache invalidation through the use of the `cachespec.xml` file. Invalidation policies can be dynamically defined in the same `cachespec.xml` that is used for configuring the cacheable objects, found inside the Web Application Archive (WAR) WEB-INF or enterprise bean WEB-INF directory. For detailed information of the `cachespec.xml` syntax, see the topic "cachespec.xml file" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Invalidation rules can be specified within a cache-entry with an `<invalidation>` tag that defines an event that must invalidate the corresponding cache entries, or with a `<timeout>` tag that configures the duration to keep the cache entries. They are defined in exactly the same manner as dependency IDs and the IDs generated by the invalidation rules are used to invalidate cache entries that have those same dependency IDs. Multiple invalidation rules can exist per cache-entry. All invalidation rules execute separately.

The following sections discuss the syntax of these invalidation rules in `cachespec.xml` and shows examples on how to apply the rules in WebSphere Commerce for the following:

- Servlet-based invalidation rules
- Command-based invalidation rules
- Timeout and priority invalidation rules

Servlet-based invalidation rules

This method of invalidation provided by WebSphere Application Server dynamic caching is to invalidate other cache entries upon execution of a servlet request.

Syntax in cachespec.xml:

```
<cache-entry>
  <class>servlet</class>
  <name>servlet_name</name>
  <invalidation>invalidation_id
    <component id="" type="pathinfo">
      ...
    </component>
  </invalidation>
</cache-entry>
```

Note:

`<class>`

A value of "servlet" indicates that the invalidation is triggered by a servlet.

`<invalidation>`

An element that is used to identify the event that triggers the invalidation and what cache entries with the specified ID to invalidate.

`pathinfo`

In WebSphere Commerce, all servlet requests go through a single controller servlet which is either:
`com.ibm.commerce.server.RequestServlet.class` or
`com.ibm.commerce.tools.common.ToolsRequestServlet.class`.

To identify a unique request URI, the "pathinfo" component type can be used to filter the pathinfo name from the request.

Example of polices defined in cachespec.xml: To invalidate the interest item list cache page identified by its reference number when an InterestItemDelete URL is issued to delete catalog entries from the list:

```
<cache-entry>
  <class>Servlet</class>
  <name>com.ibm.commerce.server.RequestServlet.class</name>

  <invalidation>listId
    <component id="" type="pathinfo" ignore-value="true">
      <required>true</required>
      <value>/InterestItemDelete</value>
    </component>
    <component id="listId" type="parameter">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>
```

Command-based invalidation rules

This method invalidates other cache entries as a side effect of running a command. It provides an event-driven mechanism to trigger the invalidation at the command level in WebSphere Commerce. Users can examine the business logic written in a command and identify specific cache objects that might be affected upon successful execution of the command and then define the invalidation rules accordingly. One criteria of such commands is that they must implement the WebSphere Commerce Programming Model and the methods and the fields in the commands that handle the input parameters must be externalized if the invalidation IDs are going to be generated based on those input parameters.

Refer to the WebSphere Application Server documentation for details of this support. See also, "Writing commands for command-based invalidation" on page 86.

Syntax in cachespec.xml:

```
<cache-entry>
  <class>command</class>
  <name>fully_qualified_class_name_of_the_command</name>
  <invalidation>invalidation_rules</invalidation>
</cache-entry>
```

Note:

<class>

A value of command indicates that the invalidation is triggered by a command.

<name>

Contains the fully qualified path of the command.

Example of polices defined in cachespec.xml: To invalidate the product display page cached by its reference number when the CatalogEntryUpdate command is invoked to change the catalog entry:

```
<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.catalogmanagement.commands.CatalogEntryUpdateCmdImpl</name>

  <invalidation>productId
    <component id="getCatentryId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>
```


Timeout and priority-based invalidation rules

The timeout mechanism using the `<timeout>` tag in a policy provides a simple way to remove any cache entries after a preset duration. The mechanism is useful when it is not feasible to set up specific invalidation rules or trigger events that are timed to remove the cache entries.

The WebSphere Application Server dynamic caching service automatically removes cache entries when the cache is full based on a modified Least Recently Used (LRU) algorithm, in which a priority weighting using the `<priority>` tag is used to decide which entries to remove from the cache. The higher the priority value for a particular cache entry relative to other cache entries, the longer this cache entry tends to stay in the cache relative to others.

Rather than removing the entries from memory when the cache becomes full, you can configure disk offload and have the entries copied onto the file system for use later. (The location is configurable.) Cache entries will also be off-loaded to disk upon server shut down and can be reused when the server is restarted. In addition, the following dynamic cache service settings also affect the invalidation process:

Cache Size

Determines the maximum number of entries the cache holds.

Default Priority

Determines by default, how long an entry stays in a full cache

Disk offload

Specifies whether disk offload is enabled, and disk offload location.

Syntax in `cachespec.xml`:

```
<cache-entry>
  .....
  <cache-id>
    <component >.....</component>
    <priority>priority_value</priority>
    <timeout>time_in_cache</timeout>
  </cache-id>
</cache-entry>
```

Example of policies defined in `cachespec.xml`: To remove the shopping cart cache content 3600 seconds after it is created or used:

```
<cache-id>
  <component id="" type="pathinfo">
    <required>true</required>
    <value>/OrderItemDisplay</value>
  </component>
  <component id="orderId" type="parameter">
    <required>true</required>
  </component>
  <timeout>3600</timeout>
</cache-id>
```

Removing cache entries through the Cache Monitor

The WebSphere Application Server Cache Monitor provides the options to manually invalidate cache entries. There exists a **Clear Cache** button that empties the entire cache contents. Each cache entry can be invalidated individually and a group of cache entries can be invalidated as a template (for example, all entries associated with the `ProductDisplay.jsp`). Groups of cache entries identified by a dependency ID may also be invalidated as a block on the monitor.

Invoking the dynamic cache invalidation APIs

WebSphere Application Server dynamic caching provides the following APIs to support programmatic invalidation:

- `com.ibm.websphere.cache.invalidateById`
- `com.ibm.websphere.cache.invalidateByTemplate`

Refer to the WebSphere Application Server documentation for details on using these APIs.

WebSphere Commerce takes advantage of these APIs, provides a homemade invalidation wrapper, `DynaCacheInvalidation` command to support a table-driven invalidation API inside WebSphere Commerce. It invalidates cache objects by processing the records in the `CACHEIVL` table. Refer to “Cache Invalidation API (`DynaCacheInvalidation`)” on page 90 for more details.

Writing commands for command-based invalidation

In order to allow a command call be intercepted by the dynamic cache, the command must be written to the WebSphere Command Framework with its implementation class extending from `CacheableCommandImpl` (in the `com.ibm.websphere.command` package). To simplify command writing for command-based invalidation, WebSphere Commerce has updated the abstract classes, `ControllerCommandImpl` and `TaskCommandImpl` to extend from `CacheableCommandImpl` so that any commands extend from these abstract classes would also extend from `CacheableCommandImpl` and therefore be eligible for command-based invalidation.

When writing such commands, it is also important to know what the invalidation IDs are, and understand the invalidation rules that intercepts calls to the commands. Since invalidation IDs are generated based on methods and fields present in the command as input parameters, all the methods that are required to construct the invalidation IDs, should be provided in the command interface and be implemented.

An example of using command invalidation in WebSphere Commerce

The following example shows how WebSphere Commerce uses command invalidation. When the command `DeleteMemberGroupMemberCmdImpl` which deletes a particular member belonging to a particular member group is executed successfully, the dynamic cache will invalidate the cache-entry defined in the invalidation rule. In this example, it is defined as “`DC_userId:userId`” where `userId` is the value being returned from the `getMemberId` method. (See “Cache servlet filtering” on page 80 for the details of caching the member group information in WebSphere Commerce.) For example, `DC_userId:-1000`, `DC_userId:-1001`, and so on. This command has a `get` method, `getMemberId()`, that retrieves the user ID that is being deleted and this method is used in computing which cache entries with a dependency ID based on a user ID gets deleted. The same logic applies for the command `AddMemberGroupMemberCmdImpl` which also has a `get` method, `getMemberId()`:

```
<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.membergroup.commands.AddMemberGroupMemberCmdImpl</name>
  <name>com.ibm.commerce.membergroup.commands.DeleteMemberGroupMemberCmdImpl</name>

  <invalidation>DC_userId
```

```

    <component type="method" id="getMemberId">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

```

Note: All the above invalidation rules are shipped with WebSphere Commerce in the sample cache policy. You can find more sample invalidation rules in the WebSphere Commerce installation directory under the `/samples/dynacache/invalidation` subdirectory. Please see the README file entitled "Sample invalidation cache policies for Dynacache" for more information on how to incorporate the invalidation rules into the `cachespec.xml` file.

Cache invalidation example

The following example shows how to set up caching policies in the `cachespec.xml` file to cache the `ProductDisplay` JSP page for the Consumer Direct business model in WebSphere Commerce and how to invalidate the cache entry by defining the invalidation rules in the same XML file. The example defines multiple dependency IDs along with the cache ID generation rule for the JSP file. Each dependency ID is used to invalidate the cache entry when the cache entry is updated under different circumstances. This example only shows a subset of policies required to invalidate the `ProductDisplay` JSP. For a complete example and detailed information, see the README file in the `WC_installdir/samples/dynacache/invalidation` directory.

```

<cache>
  <cache-entry>
    <class>servlet</class>
    <name>/FashionFlow/ShoppingArea/CatalogSection/CatalogEntrySubsection/ProductDisplay.jsp</name>
    <property name="save-attributes">>false</property>

    <!-- Cache ProductDisplay.jsp -->
    <cache-id>
      <component id="storeId" type="parameter">
        <required>true</required>
      </component>
      <component id="catalogId" type="parameter">
        <required>true</required>
      </component>
      <component id="productId" type="parameter">
        <required>true</required>
      </component>
      <component id="DC_lang" type="attribute">
        <required>true</required>
      </component>
      <component id="DC_curr" type="attribute">
        <required>true</required>
      </component>
      <component id="DC_porg" type="attribute">
        <required>true</required>
      </component>
      <component id="DC_cont" type="attribute">
        <required>true</required>
      </component>
      <component id="DC_mg" type="attribute">
        <required>true</required>
      </component>
    </cache-id>

    <!-- Used for invalidating the product display cache entry -->

```

```

<!-- that belongs to a specific store -->
<dependency-id>storeId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<!-- Used for invalidating the cache entry of a specific product -->
<dependency-id>productId
  <component id="productId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<!-- Used for invalidating the product display cache entry -->
<!-- that belongs to a specific catalog in the store -->
<dependency-id>storeId:catalogId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
  <component id="catalogId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>

<!-- Used for invalidating the product display cache entry -->
<!-- that is under a specific contract -->
<dependency-id>contractId
  <component id="DC_cont0" type="attribute">
    <required>true</required>
  </component>
</dependency-id>
</cache-entry>

<cache-entry>
  <class>command</class>
  <sharing-policy>not-shared</sharing-policy>
  <name>com.ibm.commerce.catalogmanagement.commands.AddCatalogDescCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.UpdateCatalogDescCmdImpl</name>

  <!-- ***** -->
  <!-- Invalidate all the product page cache entries that -->
  <!-- might be affected when the catalog description is changed -->
  <!-- ***** -->
  <invalidation>storeId:catalogId
    <component id="getStoreId" type="method">
      <required>true</required>
    </component>
    <component id="getCatalogId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

<cache-entry>
  <class>command</class>
  <sharing-policy>not-shared</sharing-policy>
  <name>com.ibm.commerce.catalogmanagement.commands.ListpriceAddCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.ListpriceDeleteCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.ListpriceUpdateCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.OfferAddCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.OfferDeleteCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.OfferUpdateCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.ProductAttributeUpdateCmdImpl</name>
  <name>com.ibm.commerce.catalogmanagement.commands.AttributeValueUpdateCmdImpl</name>

```

```

<name>com.ibm.commerce.catalogmanagement.commands.AddListpriceCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.DeleteListpriceCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.UpdateListpriceCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.AddOfferCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.DeleteOfferCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.UpdateOfferCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.UpdateAttributeCmdImpl</name>
<name>com.ibm.commerce.catalogmanagement.commands.UpdateAttributeValueCmdImpl</name>

<!-- ***** -->
<!-- Invalidate the specific product page cache entry when the -->
<!-- product is updated -->
<!-- ***** -->
<invalidation>productId
  <component id="getCatentryId" type="method">
    <required>true</required>
  </component>
</invalidation>
</cache-entry>

<cache-entry>
  <class>command</class>
  <sharing-policy>not-shared</sharing-policy>
  <name>com.ibm.commerce.contract.commands.ContractSuspendCmdImpl</name>
  <name>com.ibm.commerce.contract.commands.ContractTCDeployCmdImpl</name>

  <!-- ***** -->
  <!-- Invalidate all the product page cache entries under a -->
  <!-- specific contract -->
  <!-- ***** -->
  <invalidation>contractId
    <component id="getContractId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreProfileUpdateCmdImpl</name>
  <name>com.ibm.commerce.tools.devtools.flexflow.ui.commands.impl.FlexflowUpdateCmdImpl</name>
  <name>com.ibm.commerce.store.commands.StoreOpenCmdImpl</name>
  <name>com.ibm.commerce.store.commands.StoreCloseCmdImpl</name>

  <!-- ***** -->
  <!-- Invalidate all the product page cache entries that -->
  <!-- belong to the store when the store is updated -->
  <!-- ***** -->
  <invalidation>storeId
    <component id="getStoreId" type="method">
      <required>true</required>
    </component>
  </invalidation>
</cache-entry>

<cache-entry>
  <class>command</class>
  <sharing-policy>not-shared</sharing-policy>
  <name>com.ibm.commerce.catalogimport.commands.CatalogImportJobAddCmd</name>

  <!-- ***** -->
  <!-- Invalidate all the product page cache entries that -->
  <!-- belong to the store when the store catalog is updated -->
  <!-- ***** -->

```

```
<invalidation>storeId
  <component id="getStoreId" type="method">
    <required>true</required>
  </component>
</invalidation>
</cache-entry>
```

```
</cache>
```

Cache Invalidation API (DynaCacheInvalidation)

The WebSphere Commerce scheduler periodically calls the `DynaCacheInvalidation` command to process the records in the `CACHEIVL` table and then calls the dynamic cache invalidation API to invalidate the cache entries. The `startTime` is retrieved from the checkpoint file. The checkpoint file is written into the `instance_name/cache` directory. This directory gets created the first time that `DynaCacheInvalidation` is executed. The timestamp of the last record is written into the checkpoint file during the wind up stage of the command.

CACHEIVL table

The rules for the `DynaCacheInvalidation` command when processing the `CACHEIVL` table are as follows:

- If the `template` column is set, then the `DynaCacheInvalidation` command calls the dynamic cache invalidation API (`invalidateByTemplate`) and uses the name as the template ID (for example, `/webapp/wcs/stores/ToolTech/ShoppingArea/CatalogSection/CategorySubsection/StoreCatalogDisplay.jsp`).
- If the `DATA_ID` column is set and the `template` name is not set, then the `DynaCacheInvalidation` command calls the dynamic cache invalidation API (`invalidateById`) and uses the `DATA_ID` as the ID (for example, `StoreCatalogDisplay:storeId:10151`).
- When the dynamic cache invalidation API is called, it invalidates the cache entries.

Invalidation in external caches

Result caches can be pushed to external caches such as IBM HTTP Server or Edge Server. Invalidation will occur in external caches when:

- The cache is congested
- Time out occurs
- Invalidation message is sent from WebSphere Application Server

Web server plug-in cache

For WebSphere Application Server 5.0, the Web server plug-in contains a built-in Edge Side Include (ESI) processor. The ESI processor has the ability to cache whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an *in-memory* cache, not a disk cache. Therefore, the cache entries are not saved when you restart the Web server.

When a request is received by the Web server plug-in, it is sent to the ESI processor, unless the ESI processor is disabled. (It is enabled by default.) If a cache miss occurs, a `Surrogate-Capabilities` header is added to the request and the request is forwarded to WebSphere Application Server. If the dynamic servlet cache

is enabled in the application server, and the response is edgeable, the application server returns a Surrogate-Control header in the response to the WebSphere plug-in.

The ESI processor is configurable through the WebSphere Web server plug-in configuration file, `plugin-cfg.xml`. For example:

```
<Property Name="esiEnable" Value="true"/>
<Property Name="esiMaxCacheSize" Value="1024"/>
<Property Name="esiInvalidationMonitor" Value="true" />
```

where:

- `esiEnable` is used to disable the ESI processor by setting the value to false. ESI is enabled by default.
- `esiMaxCacheSize` is the maximum size of the cache in 1K bytes. The default maximum size of the cache is 1 MB. If the cache is full, the first entry to be removed from the cache is the entry that is closest to expiration. (The default expiration time for ESI cache entries is 24 hours.)
- `esiInvalidationMonitor` specifies whether or not the ESI processor should receive invalidations from the application server.

There are three methods by which the cache entries are removed from the ESI cache:

- An entry's expiration timeout is invoked.
- An entry is purged to make room for new entries.
- The application server sends an explicit invalidation for a group of entries.

In order for this mechanism to be enabled, the `esiInvalidationMonitor` must be set to `true` and the `DynaCacheEsi.ear` application which is located in the `installableApps` directory must be installed on the application server.

Note: The `DynaCacheEsi.ear` should use the `VH_instance_name` as the virtual host.

For more information on the ESI processor, see the topic "Configuring Edge Side Include caching" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Simple file servlet

By default, WebSphere Commerce does not cache static data such as images and HTML. This is set by setting the system property `com.ibm.servlet.file.esi.timeOut` to a value of 0 (zero). If you want to enable images and HTML caching, you can either:

- Set up the cache entry in the `cachespec.xml` file for the `SimpleFileServlet`. Setting up the cache entry in the `cachespec.xml` is a better approach because controlling what is being cached from WebSphere Application Server also gives you the ability to invalidate the cache entries when needed.

Note that you are storing possibly all the images as cache entries at the application server which may compromise performance when the maximum number of cache entries is reached. It is recommended that a lower priority value relative to other cache entries be set for the `SimpleFileServlet` cache entry.

- Modify the system property `com.ibm.servlet.file.esi.timeOut` setting to a non-zero timeout value.

A reason why invalidation is required for some images file is that in some business models, the logo and banner can be easily modified.

For example:

```
<cache-entry>
  <class>servlet</class>
  <name>com.ibm.ws.webcontainer.servlet.SimpleFileServlet.class</name>
  <property name="EdgeCacheable">true</property>
  <cache-id>
    <component id="" type="pathinfo">
      <required>true</required>
    </component>
  </cache-id>
  <dependency-id>
    <component id="" type="pathinfo">
      <required>true</required>
    </component>
  </dependency-id>
  <timeout>1800</timeout>
</cache-entry>
```

Here is an example of the cache entry for invalidating the images.

```
<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreLogoUpdateCmdImpl</name>
  <invalidation>FashionFlow/images/logo.gif</invalidation>
</cache-entry>

<cache-entry>
  <class>command</class>
  <name>com.ibm.commerce.tools.devtools.store.commands.StoreBannerUpdateCmdImpl</name>
  <invalidation>FashionFlow/images/banner.gif</invalidation>
</cache-entry>
```

Problem determination

If you are experiencing problems with getting your pages to be cached properly using dynamic caching, consult the following information:

- **Problem:** The Cache Monitor is not accessible.
Solution: Ensure you have regenerated the Web Server plug-in and restarted the Web server and the application server so that the new virtual host and port number mappings are loaded.

- **Problem:** No pages whatsoever are being cached.
Solution: Verify that the cachespec.xml file has been loaded into WebSphere Application Server. If the cachespec.xml is loaded, then the policies should be viewable from the Cache Monitor by clicking on the **Cache Policies** link on the side bar. In addition to the WebSphere Application Server SystemOut.log file will show a line similar to:

```
[6/6/03 21:21:12:635 EDT] 65c60609 ConfigManager I DYNA0047I: Successfully loaded
cache configuration file D:\WebSphere\AppServer5\installedApps\buzz\WC_demo.ear\
Stores.war\WEB-INF/cachespec.xml.
```

Note that the above line has been split for display purposes.

- **Problem:** Only some of your pages are being cached.
Solution: There is in all likelihood a problem with the cachespec.xml file. You can turn on the trace and then access the page under being investigation. The trace should indicate the problem.

For information on enabling tracing, see “Enabling tracing components” on page 61.

To turn on the Dynamic Cache trace (WebSphere Commerce side), the **Trace Specification** should contain

`"com.ibm.websphere.commerce.WC_CACHE=all=enabled"`.

To turn on the Dynamic Cache trace (WebSphere Application Server side) the **Trace Specification** should contain `"com.ibm.ws.cache.*=all=enabled"`.

- **Problem:** The DynaCacheInvalidation command does not invalidate.

Solution: Verify that the command is functional from the URL. If you do need to run the command from the URL, then the `cacheinvalidation.jsp` file must be in the docpath. Even if it is not in the path, the command will run however you will not receive a display page to report its completion. Once the URL invocation is correct, you can verify whether it has been scheduled to run through the WebSphere Commerce scheduler, the default is to run unscheduled.

- **Problem:** Invalidation through dependency ID does not work properly.

Solution: Verify that the dependency ID for the intended servlet or JSP file is within the same cache-entry block. If this is the case, you must ensure that multiple definitions of the same dependency ID in different cache entry blocks have the identical definition. That is, if your dependency id is defined for some cache-entry as follows, the definition of the dependency ID in any other cache-entry block must be exactly the same:

```
<dependency-id>storeId
  <component id="storeId" type="parameter">
    <required>true</required>
  </component>
</dependency-id>
```





Chapter 6. System maintenance





This chapter discusses the database cleanup utility and how you can use it to maintain the database on your system.

Database Cleanup utility

The Database Cleanup utility allows you to delete many objects from the database at the same time. You may want to do this if you have changed a lot of information in your database and have unused tables or rows. There are several objects that you can delete from the database. For more on these objects, see “Cleaning the database” on page 105.

When the Database Cleanup utility deletes an object, the records in the object’s tables are deleted to preserve the referential integrity of the database. The Database Cleanup utility command cleans the database in one of two ways: top-down or bottom-up. Top-down deletes all rows from the child tables with a delete cascade. If a delete restrict is specified in the referential integrity, the delete cascade will fail and you will have to use the bottom-up method.

    To use the bottom-up method, specify yes for the FORCE parameter in the command syntax, which first deletes the child tables, followed by the parent table.

    Another way to trigger the bottom-up method is to specify the LOGLEVEL parameter as 2 in the command syntax. Specifying 0 logs nothing, and 1 only logs the delete statements from the top table. LOGLEVEL 2 logs the delete statements from each deleted child table until the top table. Although selecting 2 triggers the bottom-up method, it cannot guarantee a successful deletion if there is a delete restrict in the referential integrity. To delete records with a delete restrict, specify the FORCE parameter as yes.

You can expect a longer response time with the bottom-up method if the table contains many child tables. For example, the MEMBER table contains more than 500 child tables. For performance reasons, you should use the top-down method.

Note: Only use Loglevel 2 if Loglevel 0 or Loglevel 1 fails. Only use the FORCE parameter set to yes if the FORCE parameter set to no fails.

The Database Cleanup utility is configurable, extensible, and adaptable. Aside from the preset cleanup configurations, you can add new objects to the database table to define which tables and rows to clean. Refer to adding a new configuration to the Database Cleanup utility.

If you have extended your database schema by creating new tables, you can use the Database Cleanup utility to clean your new tables. If you have changed your database schema (such as adding new columns to one table, changing the foreign key primary key relationship, or adding a new child table to the referential integrity path), the Database Cleanup utility will automatically adapt to the changes. If you change the column names, update the configuration data in the CLEANCONF table.

The Database Cleanup utility deletes records in child tables based on the delete rule of the referential integrity definition in the database schema. You can set the delete rule to on delete cascade, on delete set null, or on delete restrict. If you add new tables, ensure that the referential integrity and delete rule is properly defined. Otherwise, the Database Cleanup utility cannot work with your new tables.

Note: You should only run the Database Cleanup utility on a staging server to clean the staglog object. The staging database is different from the production database. The staging database only has configuration data without the operation data. Deleting configuration data might cause a delete cascade on the operation data. When the Stage Propagate utility propagates the deletion to the production database, this might cause a cascade delete to the operation data (which you want to keep). To clean configuration data, run the Database Cleanup utility on the production database.

Database Cleanup utility objects

The Database Cleanup utility refers to the CLEANCONF table to determine which tables and which rows to delete when a particular object and object type are specified. The following table describes preconfigured deletion scenarios from the CLEANCONF table. You can configure your own deletion objects by adding similar rows to the CLEANCONF table.

Object	Type	Statements
account	obsolete	delete from account where markfordelete = 1 and trdtype_id = 0 and trading_id not in (select account_id from trading) and trading_id not in (select distinct account_id from ordpymthd)
address	obsolete	delete from address where status = 'T' and (days(CURRENT_TIMESTAMP) - days(lastcreate)) >= ? and (address_id not in (select distinct address_id from orderitems where address_id is not null)) and (address_id not in (select distinct address_id from orders where address_id is not null)) and (address_id not in (select distinct allocaddress_id from orderitems where allocaddress_id is not null))
atp_inventory	obsolete	delete from receipt where qtyonhand = 0 and qtyinkits = 0 and receipt_id not in (select distinct receipt_id from ordpickhst where receipt_id is not null) and receipt_id not in (select distinct receipt_id from ordshipst where receipt_id is not null)
attachment	obsolete	delete from attachment where days(current timestamp) - days(timeupdated) >=? and attachment_id not in (select attachment_id from trdattach)
auction	retracted	delete from auction where austatus = 'R' and (days(CURRENT_TIMESTAMP) - days(updatetime)) >= ?
auction	settlement_closed	delete from auction where austatus = 'SC' and (days(CURRENT_TIMESTAMP) - days(updatetime)) >= ?
auctionlog	obsolete	delete from auctionlog where (days(CURRENT_TIMESTAMP) - days(actiontime)) >= ?
autobidlog	obsolete	delete from autobidlog where (days(CURRENT_TIMESTAMP) - days(actiontime)) >= ?

Object	Type	Statements
baseitem	obsolete	delete from baseitem where markfordelete = 1 and baseitem_id not in (select baseitem_id from catentry) and baseitem_id not in (select distinct baseitem_id from itemspc where markfordelete = 0 and itemspc_id in (select distinct itemspc_id from orderitems) or itemspc_id in (select distinct itemspc_id from oicomplst) or itemspc_id in (select distinct itemspc_id from versionspc where versionspc_id in (select distinct versionspc_id from receipt)) or itemspc_id in (select distinct itemspc_id from radetail) or itemspc_id in (select distinct itemspc_id from bkordalloc) or itemspc_id in (select distinct itemspc_id from invreserve) or itemspc_id in (select distinct itemspc_id from rmaitem) or itemspc_id in (select distinct itemspc_id from rmaitemcmp) or itemspc_id in (select distinct itemspc_id from catentry))
bidlog	obsolete	delete from bidlog where (days(CURRENT TIMESTAMP) - days(actiontime)) >= ?
+ cacheivl +	obsolete	delete from cacheivl where (days(CURRENT TIMESTAMP) - days(cacstmp)) >= ?
calculation_code	obsolete	delete from calcode where published = 2 and calcode_id not in (select distinct calcode_id from ordadjust where calcode_id is not null) and calcode_id not in (select distinct calcode_id from stencalusg where calcode_id is not null) and calcode_id not in (select distinct calcode_id from ordcalcd where calcode_id is not null) and calcode_id not in (select distinct calcode_id from ordicalcd where calcode_id is not null)
catalog_group	obsolete	delete from catgroup where markfordelete = 1
catentry	without_orderitems	delete from catentry where markfordelete = 1 and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ? and catentry_id not in (select distinct catentry_id from auction) and catentry_id not in (select distinct catentry_id from orderitems where catentry_id is not null) and catentry_id not in (select distinct catentry_id from oicomplst where catentry_id is not null) and catentry_id not in (select distinct catentry_id from rmaitem where catentry_id is not null) and catentry_id not in (select distinct catentry_id from offer where offer_id in (select distinct offer_id from orderitems))
catentry	without_orderitems- iitems	delete from catentry where markfordelete = 1 and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ? and catentry_id not in (select distinct catentry_id from auction) and catentry_id not in (select distinct catentry_id from orderitems) and catentry_id not in (select distinct catentry_id from oicomplst where catentry_id is not null) and catentry_id not in (select distinct catentry_id from iitem) and catentry_id not in (select distinct catentry_id from rmaitem where catentry_id is not null) and catentry_id not in (select distinct catentry_id from offer where offer_id in (select distinct offer_id from orderitems))

Object	Type	Statements
contract	obsolete	<ol style="list-style-type: none"> 1. delete from trading where markfordelete = 1 and trdtype_id = 1 and trading_id not in (select distinct trading_id from orderitems) and trading_id not in (select distinct trading_id from rma) and trading_id not in (select distinct trading_id from ordpaymthd) and trading_id not in (select distinct account_id from ordpaymthd) 2. delete from productset where markfordelete = 1 and productset_id not in (select distinct productset_id from tradeposcn where tradeposcn_id in (select distinct tradeposcn_id from offer where offer_id in (select distinct offer_id from orderitems))) 3. delete from tradeposcn where markfordelete = 1 and tradeposcn_id not in (select distinct tradeposcn_id from offer where offer_id in (select distinct offer_id from orderitems))
coupon_promotion	expired	delete from cppmn where days(current timestamp) - days(enddate) >=?
cpgnlog	obsolete	delete from cpgnlog
cpgnstats	obsolete	delete from cpgnstats
expected_inventory_records	obsolete	delete from ra where markfordelete = 1 and ra_id not in (select distinct ra_id from receipt, radetail where receipt.radetail_id = radetail.radetail_id)
expected_inventory_record_details	obsolete	delete from radetail where markfordelete = 1 and radetail_id not in (select distinct radetail_id from receipt)
forummsg	obsolete	delete from forummsg where msgstatus = 'D' or (days(CURRENT TIMESTAMP) - days(posttime)) >= ?
fulfillment_center	obsolete	delete from ffmcenter where markfordelete = 1 and ffmcenter_id not in (select distinct ffmcenter_id from radetail) and ffmcenter_id not in (select distinct ffmcenter_id from inventory) and ffmcenter_id not in (select distinct ffmcenter_id from rma) and ffmcenter_id not in (select distinct ffmcenter_id from orderitems) and ffmcenter_id not in (select distinct allocffmc_id from orderitems) and ffmcenter_id not in (select distinct ffmcenter_id from store) and ffmcenter_id not in (select distinct rtnffmctr_id from store) and ffmcenter_id not in (select distinct ffmcenter_id from receipt) and ffmcenter_id not in (select distinct ffmcenter_id from auction) and ffmcenter_id not in (select distinct ffmcenter_id from auctionlog)
inventory_adjustments	obsolete	delete from invadjust where days(CURRENT TIMESTAMP) - days(adjustmentdate) >= ?
inventory_adjustment_codes	obsolete	delete from invadjcode where markfordelete = 1 and invadjcode_id not in (select distinct invadjcode_id from invadjust)

Object	Type	Statements
itemspecification	obsolete	delete from itemspc where markfordelete = 1 and itemspc_id not in (select distinct itemspc_id from orderitems) and itemspc_id not in (select distinct itemspc_id from oicomplst) and itemspc_id not in (select distinct itemspc_id from versionspc where versionspc_id in (select distinct versionspc_id from receipt)) and itemspc_id not in (select distinct itemspc_id from radetail) and itemspc_id not in (select distinct itemspc_id from bkordalloc) and itemspc_id not in (select distinct itemspc_id from invreserve) and itemspc_id not in (select distinct itemspc_id from rmaitem) and itemspc_id not in (select distinct itemspc_id from rmaitemcmp) and itemspc_id not in (select distinct itemspc_id from catentry)
message	obsolete	delete from message where message_id not in (select message_id from msgmemrel) or (days(CURRENT_TIMESTAMP) - days(posttime)) >= ?
msgmemrel	obsolete	delete from msgmemrel where message_id in (select m.message_id from message ms, msgmemrel m where ms.message_id = m.message_id and (status = 'D' or ((status = 'O' or sendstat = 'S') and (days(CURRENT_TIMESTAMP) - days(posttime)) >= ?)))
order	canceled	delete from orders where status = 'X' and (days(CURRENT_TIMESTAMP) - days(lastupdate)) >= ?
order	completed	delete from orders where status = 'C' and (days(CURRENT_TIMESTAMP) - days(lastupdate)) >= ?
order	deposited	delete from orders where status = 'D' and (days(CURRENT_TIMESTAMP) - days(lastupdate)) >= ?
order	shipped	delete from orders where status = 'S' and (days(CURRENT_TIMESTAMP) - days(lastupdate)) >= ?
order	stale_guest	delete from orders where (status = 'P' or status = 'I' or status = 'W' or status = 'N') and (days(CURRENT_TIMESTAMP) - days(lastupdate)) >= ? and orders.member_id in (select distinct users_id from users where registertype = 'G') and (orders_id not in (select distinct orders_id from orderitems where inventorystatus != 'NALC' and inventorystatus is not null))
order	stale_non_guest	delete from orders where (status = 'P' or status = 'I' or status = 'W' or status = 'N') and (days(CURRENT_TIMESTAMP) - days(lastupdate)) >= ? and orders.member_id in (select distinct users_id from users where registertype != 'G') and (orders_id not in (select distinct orders_id from orderitems where inventorystatus != 'NALC' and inventorystatus is not null))
organization	obsolete	delete from member where member_id in (select orgentity_id from orgentity where orgentity_id = ?)
pastats	obsolete	delete from pastats
pcstats	obsolete	delete from pcstats
pestats	obsolete	delete from pestats
policy	obsolete	delete from policy where days(current timestamp) - days(endtime) > ? and policy_id not in (select distinct policy_id from ordpaymthd) and policy_id not in (select distinct policy_id from rma)

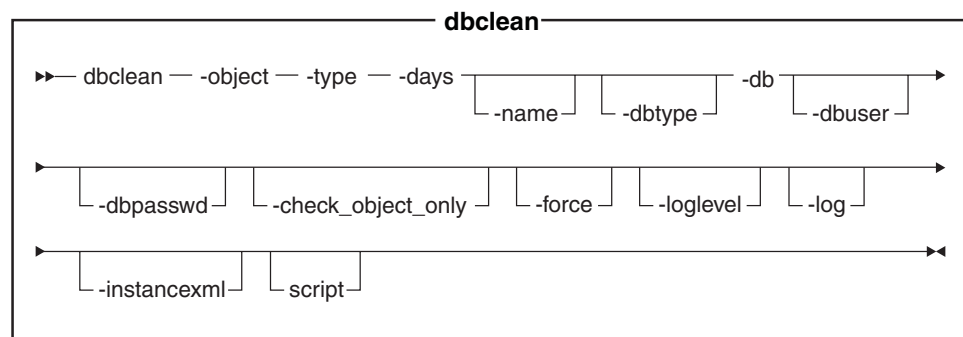
Object	Type	Statements
rfq	obsolete	delete from trading where markfordelete = 1 and trdtype_id in (2, 3, 4) and trading_id not in (select distinct trading_id from orderitems) and trading_id not in (select distinct trading_id from rma) and trading_id not in (select distinct trading_id from ordpaymthd) and trading_id not in (select distinct account_id from ordpaymthd)
rma	abandoned	delete from rma where status in ('PRC', 'EDT') and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ? and rma_id not in (select rma_id from rmaitem where rmaitem.status in ('APP', 'MAN')) and rma_id not in (select rma_id from rtnreceipt)
rma	approved_or_partly_approved	delete from rma where status = 'PND' and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ? and rma_id in (select rma_id from rmaitem where rmaitem.status in ('APP', 'MAN')) and rma_id not in (select rma_id from rtnreceipt)
rma	canceled	delete from rma where status = 'CAN' and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ?
rma	completed	delete from rma where status = 'CLO' and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ? and rma_id not in (select rma_id from rtnreceipt)
rma	not_approved	delete from rma where status = 'PND' and (days(CURRENT TIMESTAMP) - days(lastupdate)) >= ? and rma_id not in (select rma_id from rmaitem where rmaitem.status in ('APP', 'MAN')) and rma_id not in (select rma_id from rtnreceipt)
rtnreasons	obsolete	delete from rtnreason where markfordelete = 1 and rtnreason_id not in (select distinct rtnreason_id from rtnrcptdsp) and rtnreason_id not in (select distinct rtnreason_id from rmaitem)
sastats	obsolete	delete from sastats
staglog	obsolete	delete from staglog where stgprocessed = 1 and (days(CURRENT TIMESTAMP) - days(stgstmp)) >= ?
store	obsolete	delete from storeent where storeent_id = ? and type='S'
+ users	guest	delete from member where member_id in (select users_id from users where registertype='G' and (days(CURRENT TIMESTAMP) - days(prevlastsession)) >= ? And (users_id not in (select member_id from orders where status != 'Q')) and (users_id > 0) and (users_id not in (select member_id from address where address_id in (select address_id from orderitems where address_id is not null and status != 'Q') or address_id in (select allocaddress_id from orderitems where allocaddress_id is not null and status != 'Q') or address_id in (select address_id from orders where address_id is not null and status != 'Q'))))

Object	Type	Statements
users	registered	delete from member where member_id in (select users_id from users where registertype= 'R' and (days(CURRENT TIMESTAMP) - days(lastsession)) >= ? and (users_id not in (select member_id from orders where status != 'Q')) and (users_id > 0) and (users_id not in (select member_id from address where address_id in (select address_id from orderitems where address_id is not null and status != 'Q') or address_id in (select allocaddress_id from orderitems where allocaddress_id is not null and status != 'Q') or address_id in (select address_id from orders where address_id is not null and status !='Q')))))
usrtraffic	obsolete	delete from usrtraffic where (days(CURRENT TIMESTAMP) - days(stmp)) >= ?
vendor	obsolete	delete from vendor where markfordelete = 1 and vendor_id not in (select distinct vendor_id from ra) and vendor_id not in (select distinct vendor_id from receipt where vendor_id is not null)
productset	obsolete	delete from productset where markfordelete = 1 and productset_id not in (select productset_id from tradeposcn where productset_id is not null)
productset	obsolete	delete from productset where productset_id in (select productset_id from tradeposcn where productset_id is not NULL and markfordelete = 1 and type = 'C')
tradeposcn	obsolete	delete from tradeposcn where markfordelete = 1 and type = 'S'

Database Cleanup commands

Database Cleanup utility command (AIX, iSeries, Linux, Solaris and Windows 2000)

The Database Cleanup utility removes objects from your database. To run the Database Cleanup utility, type the following from a command line. Type the entire command on one line.



400 iSeries now uses `dbclean.sh`. To run shell scripts on iSeries do the following:

1. Log on with a user profile that has a CCSID other than 65535.
2. Open a QSHLL command window by typing the following command on an OS/400 command line: STRQSH.

3. Run the command as follows:
/QIBM/ProdData/CommerceServer55/bin/dbclean.sh (parameters . . .)

Note: The parameter `-dbuser` must always be specified.

z/OS If you are working with DB2 for z/OS, you need to provide some additional parameters relative to the above syntax:

- `-dbtype DB2/390`
- `-dbuser DB_user`
- `-dbpassword DB_password`
- `-dbschema DB_schema` (for DB2 for z/OS databases only)

Oracle Note:

- You must include the optional parameters, logon user ID, and password in the command even if you are currently running this utility with the same user ID.
- **Linux 400** Oracle references do not apply to the iSeries or Linux platforms.

Parameter values:

object The name of the object from which obsolete records will be deleted. Type one of the following object names:

- `account` to delete account objects.
- `address` to delete address objects.
- `atp_inventory` to delete receipt information objects.
- `attachment` to delete attachment objects.
- `auction` to delete auction objects.
- `auctionlog` to delete auction log objects.
- `autobidlog` to delete autobids objects.
- `baseitem` to delete product information objects.
- `bidlog` to delete bid log objects.
- `cachivl` to delete cached objects.
- `calculation_code` to delete calculation code objects.
- `catentry` to delete catalog entry objects.
- `catalog_group` to delete catalog group objects.
- `contract` to delete contract objects.
- `coupon_promotion` to delete coupon objects
- `cpnlog` to delete campaign objects.
- `cpnstats` to delete campaign statistic objects.
- `expected_inventory_records` to delete inventory objects.
- `expected_inventory_records_details` to inventory detail objects.
- `forummsg` to delete message objects between a Site Administrator and customers.
- `fulfillment_center` to delete fulfillment center objects.
- `inventory_adjustments` to delete inventory objects.
- `inventory_adjustment_codes` to delete inventory code objects.
- `itemspecification` to delete specified item objects.

- `jdbccustomizer` to specify the location of the customizer parameter for use with `dbclean`. To configure `dbclean` to use the toolbox `jdbc` driver with `iSeries`, you must specify `DB2/iSeries` as the `dbtype`, and your customizer file should contain:
`jdbcDriver=com.ibm.as400.access.AS400JDBCdriver`
`jdbcUrlPrefix=jdbc:as400://`. The default for `DB2` on `iSeries` is:
`jdbcDriver=com.ibm.db2.jdbc.app.DB2Driver`
`jdbcUrlPrefix=jdbc:db2://`. The default for `DB2` on other platforms is:
`jdbcDriver=COM.ibm.db2.jdbc.app.DB2Driver` `jdbcUrlPrefix=jdbc:db2`.


Note: Other combinations may work but have not been tested.

- `message` to delete auction-related message objects.
- `msgmemrel` to delete message member relationship objects.
- `orders` to delete order objects.
- `organization` to delete organization objects.
- `pastats` to delete Product Advisor statistic objects.
- `pcstats` to delete Product Comparison statistic objects.
- `pestats` to delete Product Explorer statistic objects.
- `policy` to delete policy objects.
- `product_sets` to delete product set objects.
- `rfq` to delete request for quote objects.
- `rma` to delete returned item objects.
- `rtnreasons` to delete `rtnreason` objects.
- `sastats` to delete Sales Assistant statistic objects.
- `staglog` to delete staged objects.
- `store` to delete store objects.
- `user` to delete user objects.
- `usrtraffic` to delete user traffic log objects.
- `vendor` to delete vendor objects.

type The type of object you want to delete. Refer to the individual commands under cleaning the database.

days The minimum days in existence for a record to be deleted.

name (Optional) The ID of the object to be deleted. This parameter is required if `member` was indicated as the value for the `organization` parameter and `organization` was indicated as the `type` value.

dbtype (Optional) The database type is `DB2`, `DB2/iSeries`, `DB2/390` (for `DB2` for `z/OS` databases) or Oracle databases. The default is `DB2`.  400 The default is `DB2/iSeries`.


db The name of the database.  400 The name of the database as found in the relational database directory.

Note: If you have configured the `jdbccustomizer` parameter to use the toolbox `jdbc` driver, you must specify the hostname of the machine where the database resides instead of the database name, for example, `-db hostname.ibm.com`.

 Oracle

Use *host:port:sid*. For example, *myhost:1521:mydb*.

dbuser

The logon ID of the administrator who has created the schema or Site Administrator of the database. If this parameter is not specified, the ID of the user invoking the utility is used.  (Required) The user profile associated with the WebSphere Commerce instance. This is also the schema name.

dbpasswd

(Optional) The password of the logon ID that is specified by the *dbuser* parameter for the DB2 database. If not specified, the system prompts you to enter the password.

dbscheme

The DB2 schema name for a DB2 for z/OS (DB2/390) database found in Linux Hybrid environments.

check_object_only

(Optional) The Database Cleanup utility lists all child tables, which might be impacted by the database cleanup, and delete restricts if you specify *yes*. The utility does not perform a check if you leave the parameter to *no* (the default).

force (Optional) The *force* option can be set to *yes* or *no*. If you specify *yes*, the utility deletes the child tables, followed by the parent table.


loglevel


(Optional) The level of logging to be performed during the database cleanup. If no value is specified, the default logging level is 0.

- Type 0 to specify that no log activities are recorded. This is the default.
- Type 1 to specify that DELETE statements are to be recorded only for the table specified.
- Type 2 to specify that DELETE statements are to be recorded for the table specified, and for any child tables. Loglevel 2 forces the Database Cleanup utility to use the bottom-up method.

log (Optional) The path and name of the log file in which the utility records its activities. The issuer of this command must have write authority to the specified path and the path must already exist. If this parameter is not specified, a log file called *dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log* is created in the following log directory.

   *WC_installdir/logs*

 *WC_installdir\logs*

 If this parameter is not specified, a log file called *dbclean_dbuser_yyyy.mm.dd_hh.mm.ss.zzz.log* is created in the *WC_userdir/instances* directory.

instancexml

Used when deleting a store. *instancexml* takes the value of the absolute file name in the form of *WC_installdir/instances/instancexml/xml/instancexml.xml*.








Note: This parameter is required for some objects that own file assets, such as user, organization, store, and attachment.

script (Optional) Specifies the path and name of the output script to be saved. By default, it is saved as

```
Windows WC_installdir\logs\assetclean_userName_timestamp_.cmd
400 /WC_userdir/instances/assetclean_userName_timestamp_.sh
and WC_installdir/logs/assetclean_userName_timestamp_.sh for other
platforms.
```

Cleaning the database

To delete unused or obsolete objects from the WebSphere Commerce database using the Database Cleanup utility, select one of the following:

- Account objects
- Address objects
- Available-to-promise (ATP) inventory objects
- Attachment objects
-  Auction objects
-  Auction log objects
-  Autobid log objects
- Base item objects
-  Bid log objects
- Calculation code objects
- Campaign objects
- Campaign statistic objects
- Catalog entry objects
- Catalog group objects
- Contract objects
- Coupon objects
- Expected inventory record objects
- Expected inventory record details object
- Forum message objects
- Fulfillment center objects
- Inventory adjustment code objects
- Inventory adjustment objects
- Item specification objects
- Member message objects
- Message objects
- Order objects
- Organization objects
-  Product Advisor statistic objects
-  Product Comparison statistic objects
-  Product Explorer statistic objects
- Policy objects
- Product set objects
- Request for quote (RFQ) objects
- Returned item objects
- rtnreason objects

- **Professional** Sales Assistant statistic objects
- Staged objects
- Store objects
- User objects
- User traffic log objects
- Vendor objects

Note: Before you can use the Database Cleanup utility, you must configure the database and set the PATH environment variables.

Setting the PATH environment variables for WebSphere Commerce utilities

Note: **400** This section is not applicable to the iSeries platform. Before using the Database Cleanup utility or the staging server, ensure that you set the following in the PATH environment variables:

- **2000** **DB2**
WC_installdir/bin and *WC_installdir\sqllib\bin*.
- **2000** **Oracle**
WC_installdir\bin and *WC_installdir\ora9i\bin*
- **2000** **DB2**
WC_installdir\bin and *WC_installdir\sqllib\bin*
- **2000** **Oracle**
WC_installdir\bin and *WC_installdir\ora9i\bin*
- **AIX** **DB2**
WC_installdir/bin and *WC_installdir/lpp/db2_08_01*
- **AIX** **Oracle**
WC_installdir/bin and */usr/ora9i/bin*
- **Solaris** **DB2**
WC_installdir/bin and *WC_installdir/IBMDB2/V8.1*
- **Linux** **DB2**
WC_installdir/bin and *opt/IBM/db2/V8.1*
- **Solaris** **Oracle**
WC_installdir/bin and *WC_installdir/ora9i/bin*

Deleting objects











This section covers deleting specific unused or obsolete objects from the WebSphere Commerce database using the Database Cleanup utility.

Deleting account objects

To delete account objects, do the following:

1. Type the following:

- **Windows** **DB2**
`dbclean -object account -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`

-  
`dbclean -object account -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object account -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object account -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on account objects, see “Examples of deleting objects” on page 129.

Deleting address objects

To delete address objects, do the following:

1. Type the following:

-  
`dbclean -object address -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object address -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object address -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object address -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.





2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting address objects, see “Examples of deleting objects” on page 129.

Deleting available to promise inventory objects

To delete available-to-promise (ATP) inventory objects, do the following:

1. Type the following:

-  
`dbclean -object atp_inventory -type typename -db dbname -loglevel
loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object atp_inventory -type typename -db dbname -logLevel
loglevel -dbtype oracle -dbuser user -dbpasswd password`

- ▶ AIX
 - ▶ 400
 - ▶ Solaris
 - ▶ Linux
 - ▶ DB2

```
. dbclean.sh -object atp_inventory -type typename -db dbname -loglevel loglevel -dbuser user -dbpasswd password
```
- ▶ AIX
 - ▶ Solaris
 - ▶ Oracle

```
. dbclean.sh -object atp_inventory -type typename -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting ATP inventory objects, see “Examples of deleting objects” on page 129.

Deleting attachment objects

To delete attachment objects, do the following:

- Type the following:

- ▶ Windows
 - ▶ DB2

```
dbclean -object attachment -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password
```
- ▶ Windows
 - ▶ Oracle

```
dbclean -object attachment -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```
- ▶ AIX
 - ▶ 400
 - ▶ Solaris
 - ▶ Linux
 - ▶ DB2

```
. dbclean.sh -object attachment -type obsolete -db dbname -dbuser user -days daysold -loglevel loglevel -dbuser user -dbpasswd password
```
- ▶ AIX
 - ▶ Solaris
 - ▶ Oracle

```
. dbclean.sh -object attachment -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

Note: The `-type` of attachment can be `markfordelete` as well. In this case, `-instancexml` is required. For example:

```
dbclean.sh -object attachment -type markfordelete -instancexml /WC_userdir/instances/instance/xml/instance.xml -db dbname -dbuser instance -loglevel
```

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting attachment objects, see “Examples of deleting objects” on page 129.











Deleting auction objects

To delete auction objects, do the following:

- Type the following:

- ▶ Windows
 - ▶ DB2

```
dbclean -object auction -type typename -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password
```


-  
`dbclean -object auction -type typename -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object auction -type typename -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object auction -type typename -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`. **Note:** For the `-type` parameter, you can specify `settlement_closed` to indicate a completed auction record, or `retracted` to indicate a retracted auction.













2. Examine the `dbclean.log_yyyy.mm.dd_hh.mm.ss.zzz` file to verify that the command was successful.

For additional examples on deleting auction objects, see “Examples of deleting objects” on page 129.

Deleting auction log objects

To delete auction log objects, do the following:

1. Type the following:

-  
`dbclean -object auctionlog -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object auctionlog -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object auctionlog -type obsolete -db dbname -days
daysold -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object auctionlog -type obsolete -db dbname -dbuser user
 -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd
password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.



2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting auction log objects, see “Examples of deleting objects” on page 129.

Deleting autobid log objects

To delete autobid log objects, do the following:

1. Type the following:

-  
`dbclean -object autobidlog -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`

- Windows Oracle
`dbclean -object autobidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object autobidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
- AIX Solaris Oracle
`. dbclean.sh -object autobidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting autobid log objects, see “Examples of deleting objects” on page 129.

Deleting base item objects

To delete base item objects, which contain information about a general family of merchandise with the same name and description, do the following:

- Type the following:

- Windows DB2
`dbclean -object baseitem -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
- Windows Oracle
`dbclean -object baseitem -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object baseitem -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
- AIX Solaris Oracle
`. dbclean.sh -object baseitem -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting base item objects, see “Examples of deleting objects” on page 129.

Deleting bid log objects

To delete bid log objects, do the following:

- Type the following:

- Windows DB2
`dbclean -object bidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`

- Windows Oracle
`dbclean -object bidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object bidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
- AIX Solaris Oracle
`. dbclean.sh -object bidlog -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting bid log objects, see “Examples of deleting objects” on page 129.

Deleting cacheivl objects

To delete cacheivl objects, do the following:

- Type the following:

- Windows DB2
`dbclean -object cacheivl -type obsolete -db dbname -days daysold -loglevel loglevel`
- Windows Oracle
`dbclean -object cacheivl -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object cacheivl -type obsolete -db dbname -dbuser user -days daysold -loglevel loglevel`
- AIX Solaris Oracle
`. dbclean.sh -object cacheivl -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting cached objects, see “Examples of deleting objects” on page 129.

Deleting calculation code objects

To delete calculation code objects, do the following:

- Type the following:

- Windows DB2
`dbclean -object calculation_code -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
- Windows Oracle
`dbclean -object calculation_code -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

- AIX
 - 400
 - Solaris
 - Linux
 - DB2
 - . dbclean.sh -object calculation_code -type obsolete -db *dbname* -loglevel *loglevel* -dbuser *user* -dbpasswd *password*
- AIX
 - Solaris
 - Oracle
 - . dbclean.sh -object calculation_code -type obsolete -db *dbname* -loglevel *loglevel* -dbtype oracle -dbuser *user* -dbpasswd *password*

Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

- Examine the dbclean_YYYY.MM.DD_HH.MM.SS.ZZZ.log file to verify that the command was successful.

For additional examples on calculation code objects, see “Examples of deleting objects” on page 129.

Deleting campaign objects

To delete campaign objects, do the following:

- Type the following:

- Windows
 - DB2
 - dbclean -object cpnlog -type all -db *dbname* -loglevel *loglevel* -dbuser *user* -dbpasswd *password*
- Windows
 - Oracle
 - dbclean -object cpnlog -type all -db *dbname* -loglevel *loglevel* -dbtype oracle -dbuser *user* -dbpasswd *password*
- AIX
 - 400
 - Solaris
 - Linux
 - DB2
 - . dbclean.sh -object cpnlog -type all -db *dbname* -loglevel *loglevel* -dbuser *user* -dbpasswd *password*
- AIX
 - Solaris
 - Oracle
 - . dbclean.sh -object cpnlog -type all -db *dbname* -loglevel *loglevel* -dbtype oracle -dbuser *user* -dbpasswd *password*

Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

- Examine the dbclean_YYYY.MM.DD_HH.MM.SS.ZZZ.log file to verify that the command was successful.

For additional examples on deleting campaign objects, see “Examples of deleting objects” on page 129.

Deleting campaign statistic objects

To delete campaign statistic objects, do the following:

- Type the following:

- Windows
 - DB2
 - dbclean -table cpnstats -type all -db *dbname* -loglevel *loglevel* -dbuser *user* -dbpasswd *password*
- Windows
 - Oracle
 - dbclean -table cpnstats -type all -db *dbname* -loglevel *loglevel* -dbtype oracle -dbuser *user* -dbpasswd *password*
- AIX
 - 400
 - Solaris
 - Linux
 - DB2
 - . dbclean.sh -object cpnstats -type all -db *dbname* -loglevel *loglevel* -dbuser *user* -dbpasswd *password*

- ▶ AIX
▶ Solaris
▶ Oracle

```
. dbclean.sh -object cpgnstats -type all -db dbname -loglevel loglevel
-dbtype oracle -dbuser user -dbpasswd password
```

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting campaign statistic records from the CPGNSTATS table, see “Examples of deleting objects” on page 129.

Deleting catalog entry objects

To delete catalog entry objects, do the following:

- Type the following:

- ▶ Windows
▶ DB2

```
dbclean -object catentry -type typename -db dbname -days daysold
-loglevel loglevel -dbuser user -dbpasswd password
```
- ▶ Windows
▶ Oracle

```
dbclean -object catentry -type typename -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```
- ▶ AIX
▶ 400
▶ Solaris
▶ Linux
▶ DB2

```
. dbclean.sh -object catentry -type typename -db dbname -days daysold
-loglevel loglevel -dbuser user -dbpasswd password
```
- ▶ AIX
▶ Solaris
▶ Oracle

```
. dbclean.sh -object catentry -type typename -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

Note: For the `-type` parameter, you can specify `without_orderitems` to indicate catalog entries which are not referenced to any order item or `without_orderitems_items` to indicate catalog entries which are not referenced to any order item or interest list item.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting catalog entry objects, see “Examples of deleting objects” on page 129.

Deleting catalog group objects









To delete catalog group objects, do the following:

- Type the following:

- ▶ Windows
▶ DB2

```
dbclean -object catalog_group -type obsolete -db dbname -loglevel
loglevel -dbuser user -dbpasswd password
```
- ▶ Windows
▶ Oracle

```
dbclean -object catalog_group -type obsolete -db dbname -loglevel
loglevel -dbtype oracle -dbuser user -dbpasswd password
```

-     
`. dbclean.sh -object catalog_group -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object catalog_group -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting catalog group objects, see “Examples of deleting objects” on page 129.





Deleting contract objects

To delete contract objects, do the following:

1. Type the following:

-  
`dbclean -object contract -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object contract -type obsolete -db database -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object contract -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object contract -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

Note:      For the `-dbor DATABASE` parameter, you must specify one of three database tables: `productset`, `tradeposn`, or `trading`.



2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting contract objects, see “Examples of deleting objects” on page 129.

Deleting coupon objects

To delete coupon objects, do the following:

1. Type the following:

-  
`dbclean -object coupon_promotion -type expired -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`

- Windows Oracle
`dbclean -object coupon_promotion -type expired -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object coupon_promotion -type expired -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
- AIX Solaris Oracle
`. dbclean.sh -object coupon_promotion -type expired -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting coupon objects, see “Examples of deleting objects” on page 129.

Deleting expected inventory record objects

To delete expected inventory record objects, do the following:

- Type the following:

- Windows DB2
`dbclean -object expected_inventory_records -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
- Windows Oracle
`dbclean -object expected_inventory_records -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object expected_inventory_records -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
- AIX Solaris Oracle
`. dbclean.sh -object expected_inventory_records -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting expected inventory record objects, see “Examples of deleting objects” on page 129.

Deleting details about expected inventory record objects

To delete details about expected inventory record objects, do the following:

- Type the following:

- Windows DB2

```
dbclean -object expected_inventory_record_details -type obsolete -db
dbname -loglevel loglevel -dbuser user -dbpasswd password
```
- Windows Oracle

```
dbclean -object expected_inventory_record_details -type obsolete -db
dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd
password
```
- AIX 400 Solaris Linux DB2

```
. dbclean.sh -object expected_inventory_record_details -type obsolete
-db dbname -loglevel loglevel -dbuser user -dbpasswd password
```
- AIX Solaris Oracle

```
. dbclean.sh -object expected_inventory_record_details -type obsolete
-db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd
password
```

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting details about expected inventory record objects, see “Examples of deleting objects” on page 129.

Deleting forum message objects

To delete forum message objects, do the following:

- Type the following:

- Windows DB2

```
dbclean -object forummsg -type obsolete -db dbname -days daysold
-loglevel loglevel -dbuser user -dbpasswd password
```
- Windows Oracle

```
dbclean -object forummsg -type obsolete -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```
- AIX 400 Solaris Linux DB2

```
. dbclean.sh -object forummsg -type obsolete -db dbname -days daysold
-loglevel loglevel -dbuser user -dbpasswd password
```
- AIX Solaris Oracle

```
. dbclean.sh -object forummsg -type obsolete -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
```

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.













- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting forum message objects, see “Examples of deleting objects” on page 129.

Deleting fulfillment center objects

To delete fulfillment center objects, do the following:

- Type the following:

-  
`dbclean -object fulfillment_center -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object fulfillment_center -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object fulfillment_center -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object fulfillment_center -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting fulfillment center objects, see “Examples of deleting objects” on page 129.

Deleting inventory code adjustment objects

To delete inventory code adjustment objects, do the following:

1. Type the following:

-  
`dbclean -object inventory_adjustment_codes -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object inventory_adjustment_codes -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object inventory_adjustment_codes -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object inventory_adjustment_codes -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting inventory code adjustment objects, see “Examples of deleting objects” on page 129.

Deleting inventory adjustment objects

To delete inventory adjustment objects, do the following:

1. Type the following:

-  
`dbclean -object inventory_adjustments -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object inventory_adjustments -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object inventory_adjustments -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object inventory_adjustments -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting inventory adjustment objects, see “Examples of deleting objects” on page 129.

Deleting specified item information objects

To delete specified item information objects, do the following:

1. Type the following:

-  
`dbclean -object itemspecification -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object itemspecification -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object itemspecification -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object itemspecification -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on specified item information objects, see “Examples of deleting objects” on page 129.

Deleting member message relationship objects

To delete member message relationship objects, do the following:

1. Type the following:

-  
`dbclean -object msgmemrel -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object msgmemrel -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object msgmemrel -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object msgmemrel -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example,
 myhost:1521:mydb.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting message member relationship objects, see “Examples of deleting objects” on page 129.

Deleting message objects

To delete message objects, do the following:

1. Type the following:

-  
`dbclean -object message -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object message -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object message -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object message -type obsolete -db dbname -days daysold
 -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example,
 myhost:1521:mydb.



2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting message objects, see “Examples of deleting objects” on page 129.

Deleting order objects

To delete order objects, do the following:

1. Type the following:

-  
`dbclean -object order -type typename -db dbname -days daysold
 -loglevel loglevel -dbuser user -dbpasswd password`

- Windows Oracle
`dbclean -object order -type typename -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object order -type typename -db dbname -dbuser user -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
- AIX Solaris Oracle
`. dbclean.sh -object order -type typename -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.

Note: For the `-type` parameter, you can specify one of six different types: `completed` to indicate a completed order, `canceled` to indicate a canceled order, `shipped` to indicate a shipped order, `deposited` to indicate an order which has been deposited, `stale_guest` to indicate stale orders from guest customers, or `stale_non_guest` to indicate stale orders by customers who are not guests.

- Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting order objects, see “Examples of deleting objects” on page 129.

Deleting organization objects

To delete organization objects, do the following:

- Type the following:

- Windows DB2
`dbclean -object organization -type specified -db dbname -dbuser user -dbpasswd password -loglevel loglevel -name organizationid -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
- Windows Oracle
`dbclean -object organization -type specified -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password -name organizationid -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
- AIX 400 Solaris Linux DB2
`. dbclean.sh -object organization -type specified -db dbname -dbuser user -dbpasswd password -loglevel loglevel -name organizationid -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
- AIX Solaris Oracle
`. dbclean.sh -object organization -type specified -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password -name organizationid -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`. **Note:** For the `-type` parameter, you can specify `organization` to indicate any organization record.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting organization objects, see “Examples of deleting objects” on page 129.

Deleting Product Advisor statistic objects

To delete Product Advisor statistic objects, do the following:

1. Type the following:

-  
`dbclean -object pastats -type all -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object pastats -type all -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object pastats -type all -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object pastats -type all -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting Product Advisor statistic objects, see “Examples of deleting objects” on page 129.

Deleting Product Comparison statistic objects

To delete Product Comparison (Product Advisor) statistic objects, do the following:

1. Type the following:

-  
`dbclean -object pcstats -type all -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object pcstats -type all -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object pcstats -type all -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object pcstats -type all -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use `host:port:sid` for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting Product Comparison (Product Advisor) statistic objects, see “Examples of deleting objects” on page 129.

Deleting Product Explorer statistic objects

To delete Product Explorer (Product Advisor) statistic objects, do the following:

1. Type the following:

-  
`dbclean -object pestats -type all -db dbname -loglevel loglevel
-dbuser user -dbpasswd password`
-  
`dbclean -object pestats -type all -db dbname -loglevel loglevel
-dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object pestats -type all -db dbname -loglevel loglevel
-dbuser user -dbpasswd password`
-   
`. dbclean.sh -object pestats -type all -db dbname -loglevel loglevel
-dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example,
myhost:1521:mydb.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting Product Explorer (Product Advisor) statistic objects, see “Examples of deleting objects” on page 129.

Deleting policy objects

To delete policy objects, do the following:

1. Type the following:

-  
`dbclean -object policy -type obsolete -db dbname -days daysold
-loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object policy -type obsolete -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object policy -type obsolete -db dbname -days daysold
-loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object policy -type obsolete -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example,
myhost:1521:mydb.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting policy objects, see “Examples of deleting objects” on page 129.

Deleting product set objects

To delete product set objects, do the following:

1. Type the following:

-  
`dbclean -object product_sets -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object product_sets -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object product_sets -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object product_sets -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on product set objects, see “Examples of deleting objects” on page 129.

Deleting request-for-quote objects

To delete request-for-quote (RFQ) objects, do the following:

1. Type the following:

-  
`dbclean -table rfq -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object rfq -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object rfq -type obsolete -db dbname -dbuser user -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object rfq -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting RFQ objects, see “Examples of deleting objects” on page 129.

Deleting returned item objects

To delete returned item objects, do the following:

1. Type the following:

-  
`dbclean -object rma -type typename -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object rma -type typename -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object rma -type typename -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object rma -type typename -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, `myhost:1521:mydb`.

Note: For the `-type` parameter, you can specify `abandoned` to indicate an abandoned record, `canceled` to indicate a canceled record, `not_approved` to indicate a rejected record, `approved_or_partly_approved` to indicate an approved or partly approved record, or `completed` to indicate a completed record.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting returned item objects, see “Examples of deleting objects” on page 129.

Deleting return reason objects

To delete the reasons for customer dissatisfaction with merchandise, or simply return reason objects, do the following:

1. Type the following:

-  
`dbclean -object rtnreason -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object rtnreason -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object rtnreason -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object rtnreason -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, `myhost:1521:mydb`.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting return reason objects, see “Examples of deleting objects” on page 129.

Deleting Sales Assistant statistic objects

To delete Sales Assistant (Product Advisor) statistic objects, do the following:

1. Type the following:

-  
dbclean -object sastats -type all -db *dbname* -loglevel *loglevel*
-dbuser *user* -dbpasswd *password*
-  
dbclean -object sastats -type all -db *dbname* -loglevel *loglevel*
-dbtype oracle -dbuser *user* -dbpasswd *password*
-     
. dbclean.sh -object sastats -type all -db *dbname* -loglevel *loglevel*
-dbuser *user* -dbpasswd *password*
-   
. dbclean.sh -object sastats -type all -db *dbname* -loglevel *loglevel*
-dbtype oracle -dbuser *user* -dbpasswd *password*

Use *host:port:sid* for the Oracle database name. For example,
myhost:1521:mydb.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting Sales Assistant (Product Advisor) statistic objects, see “Examples of deleting objects” on page 129.

Deleting staged objects

To delete staged objects, do the following:

1. Type the following:

-  
dbclean -object staglog -type obsolete -db *dbname* -days *daysold*
-loglevel *loglevel* -dbuser *user* -dbpasswd *password*
-  
dbclean -object staglog -type obsolete -db *dbname* -dbuser *user* -days
daysold -loglevel *loglevel* -dbtype oracle -dbuser *<user>* -dbpasswd
password
-     
. dbclean.sh -object staglog -type obsolete -db *dbname* -days *daysold*
-loglevel *loglevel* -dbuser *user* -dbpasswd *password*
-   
. dbclean.sh -object staglog -type obsolete -db *dbname* -days *daysold*
-loglevel *loglevel* -dbtype oracle -dbuser *user* -dbpasswd *password*

Use *host:port:sid* for the Oracle database name. For example,
myhost:1521:mydb.













2. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting staged objects, see “Examples of deleting objects” on page 129.

Deleting store objects

To delete store objects, do the following:

1. Type the following:

-  
`dbclean -object store -type specified -db dbname -loglevel loglevel
-dbuser user -dbpasswd password -name storeid -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
-  
`dbclean -object store -type specified -db dbname -loglevel loglevel
-dbtype oracle -dbuser user -dbpasswd password -name storeid
-instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
-     
`. dbclean.sh -object store -type specified -db dbname -dbuser user
-dbpasswd password -loglevel loglevel -name storeid -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
-   
`. dbclean.sh -object store -type specified -db dbname -loglevel
loglevel -dbtype oracle -dbuser user -dbpasswd password -name storeid
-instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`

Use *host:port:sid* for the Oracle database name. For example,
myhost:1521:mydb.













2. Examine the `dbclean_yyyymm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting store objects, see “Examples of deleting objects” on page 129.

Deleting user objects

To delete user objects, do the following:

1. Type the following:

-  
`dbclean -object user -type typename -db dbname -days daysold -loglevel
loglevel -dbuser user -dbpasswd password -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
-  
`dbclean -object user -type typename -db dbname -days daysold -loglevel
loglevel -dbtype oracle -dbuser user -dbpasswd password -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
-     
`. dbclean.sh -object user -type typename -db dbname -dbuser user
-dbpasswd password -days daysold -loglevel loglevel -instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`
-   
`. dbclean.sh -object user -type typename -db dbname -days daysold
-loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password
-instancexml
WC_installdir/instances/INSTANCE_NAME/xml/INSTANCE_NAME.xml`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

Note: For the *-type* parameter, you can specify *guest* to indicate a guest customer or *registered* to indicate a registered customer.













2. Examine the *dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log* file to verify that the command was successful.

For additional examples on deleting user objects, see “Examples of deleting objects” on page 129.

Deleting user traffic log objects

To delete user traffic log objects, do the following:

1. Type the following:

-  
`dbclean -object usrtraffic -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object usrtraffic -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object usrtraffic -type obsolete -db dbname -days daysold -loglevel loglevel -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object usrtraffic -type obsolete -db dbname -days daysold -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`

Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.










2. Examine the *dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log* file to verify that the command was successful.

For additional examples on deleting user traffic log objects, see “Examples of deleting objects” on page 129.

Deleting vendor objects

To delete vendor objects, do the following:

1. Type the following:

-  
`dbclean -object vendor -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`
-  
`dbclean -object vendor -type obsolete -db dbname -loglevel loglevel -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object vendor -type obsolete -db dbname -loglevel loglevel -dbuser user -dbpasswd password`

- AIX Solaris Oracle
 . dbclean.sh -object vendor -type obsolete -db *dbname* -loglevel *loglevel* -dbtype oracle -dbuser *user* -dbpasswd *password*

Use *host:port:sid* for the Oracle database name. For example, myhost:1521:mydb.

- Examine the dbclean_YYYY.MM.DD_HH.MM.SS.ZZZ.log file to verify that the command was successful.

For additional examples on deleting vendor objects, see “Examples of deleting objects” on page 129.

Adding a new configuration to the Database Cleanup utility

To add a new configuration to the Database Cleanup utility, use the following syntax as a reference. For example, object o1 consists of table R1, which contains the following columns: col1, col2, lastupdate, and col3. To configure the Database Cleanup utility to delete all objects with col1 > 10, and where lastupdate is n days ago, do the following:

DB2

- Windows AIX Solaris Linux Open a DB2 command prompt.
- Type the following:
 db2 insert into cleanconf (objectname, type, statement, namearg, sequence, daysarg) values ('o1', 'obsolete', 'delete from r1 where col1 > 10 and (days(CURRENT TIMESTAMP) - days(lastupdate)) > ?', 'no', 1, 'yes')
- 400 Run the following SQL statement:
 insert into cleanconf (objectname, type, statement, namearg, sequence, daysarg) values ('o1', 'obsolete', 'delete from r1 where col1 > 10 and (days(CURRENT TIMESTAMP) - days(lastupdate)) > ?', 'no', 1, 'yes')

Oracle











- Open an SQLPlus command window.
- Type the following:
 insert into cleanconf (objectname, type, statement, namearg, sequence, daysarg) values ('o1', 'obsolete', 'delete from r1 where col1 > 10 and (sysdate - lastupdate) > ?', 'no', 1, 'yes')

where ? is replaced by the -days parameter from the following command line. The 'no' indicates that the name parameter is not used in the statement. The 'yes' indicates that the -days parameter is used in the statement. 'obsolete' describes the cleanup type for object o1. You can use other words, but you must use the same word in the -type argument when you invoke the Database Cleanup utility command.

Example

To invoke the Database Cleanup utility command to clean the records which have been in existence for two days from the new table, type the following:

- Windows DB2
 dbclean -object o1 -db *dbname* -type obsolete -days 2 -loglevel 1













-  
`dbclean -object o1 -db dbname -type obsolete -days 2 -loglevel 1 -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object o1 -db dbname -dbuser user -type obsolete -days 2 -loglevel 1`
-   
`. dbclean.sh -object o1 -db dbname -type obsolete -days 2 -loglevel 1 -dbtype oracle -dbuser user -dbpasswd password`

Note: For the Oracle *dbname* parameter, use *host:port:sid*. For example, *myhost:1521:mydb*.













Examples of deleting objects

The following are examples of deleting objects from the database tables using optional parameters from the Database Cleanup utility command. Refer the Database Cleanup utility command for detailed parameter information.













Example 1: To verify which tables specify the delete restrict parameter, type the following:

-  
`dbclean -object objectname -type typename -db database -check_object_only yes -dbuser user -dbpasswd password`
-  
`dbclean -object objectname -type typename -db host:port:sid -check_object_only yes -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object objectname -type typename -db database -check_object_only yes -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object objectname -type typename -db host:port:sid -check_object_only yes -dbtype oracle -dbuser user -dbpasswd password`

Example 2 To use the force option on tables which specified the delete restrict parameter, type the following:

-  
`dbclean -object objectname -type typename -db database -days daysold -loglevel loglevel -force yes -dbuser user -dbpasswd password`
-  
`dbclean -object objectname -type typename -db host:port:sid -days daysold -loglevel loglevel -force yes -dbtype oracle -dbuser user -dbpasswd password`
-     
`. dbclean.sh -object objectname -type typename -db database -days daysold -loglevel loglevel -force yes -dbuser user -dbpasswd password`
-   
`. dbclean.sh -object objectname -type typename -db host:port:sid -days daysold -loglevel loglevel -force yes -dbtype oracle -dbuser user -dbpasswd password`

Example 3 The default log file name is always a variation of `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log`. To specify the log file name, type the following:

-  
`dbclean -object objectname -type typename -db database -days daysold -loglevel loglevel -log logfile`
-  
`dbclean -object objectname -type typename -db host:port:sid -days daysold -loglevel loglevel -log logfile -dbtype oracle -dbuser user -dbpasswd <password>`
-     
`. dbclean.sh -object objectname -type typename -db database -dbuser user -days daysold -loglevel loglevel -log logfile`
-   
`. dbclean.sh -object objectname -type typename -db host:port:sid -days daysold -loglevel loglevel -log logfile -dbtype oracle -dbuser user -dbpasswd <password>`

Chapter 7. Performance

WebSphere Commerce is a complex interaction between a number of products. Each product has its own performance characteristics and within the interaction of the various components, there are a number of places where performance can be affected by incorrect configuration or insufficient resources. Performance objectives include handling the following types of requests in a timely manner:

- Handling multiple customer requests
- Accessing data in the WebSphere Commerce database
- Formatting data as Web pages
- Returning responses to the shopper's browser

To optimize WebSphere Commerce, consider the following components:

- Hardware

Make sure your machine meets the minimum machine requirements as documented in the *WebSphere Commerce Installation Guide*.

In a production environment with a lot of concurrent users, multiple CPUs will help increase performance. Using a faster CPU will generally speed up most operations.

- Database

For DB2 Database tuning, please refer to the RedBook: *DB2/UDB WebSphere Performance Tuning Guide* (available from <http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg246417.html?open>).

Make sure that the maximum database connection pool size is sufficient to handle all concurrent tasks (for example, HTTP connection, scheduler threads, and so on).

- WebSphere Commerce

Make sure the server is I/O Bound - the WebSphere Commerce system performance may be impacted if a lot of file access or network access is occurring. For example, if all logging and tracing is turned on, the system could spend most of the time writing data to the disk instead of handling the workload.

Use dynamic caching as documented in Chapter 5, "Dynamic caching," on page 75

When you are using server-based session management refer to the guidelines in the *WebSphere Application Server 5.0 Tuning Guide*. You can find this Guide at the WebSphere Application Server Information Center

(<http://www.ibm.com/software/webservers/appserv/infocenter.html>). Expand **Monitor and Troubleshooting** —> **Performance** —> **Tuning Performance**.

- WebSphere Application Server

Use the *WebSphere Application Server 5.0 Tuning Guide* as a guideline on how to tune the system. You can find this Guide at the WebSphere Application Server Information Center

(<http://www.ibm.com/software/webservers/appserv/infocenter.html>). Expand **Monitor and Troubleshooting** —> **Performance** —> **Tuning Performance**.

- Other considerations:

- WebSphere Datasource (Minimum and Maximum Connection pool size, Statement Cache Size)

- Web site design
- Security (configuration, time outs, authentication, and access control)
- Web server issues (process handling, resource usage, fast response cache accelerator)
- WebSphere engine issues (Java Virtual Machine or JVM, transport queue, the caching of JSP files, EJB container)
- WebSphere Commerce session management (caching, storing sessions in memory or storing sessions in the database)
- WebSphere Application Server session management, (setting for the in-memory session count, allow overflow, timeout interval, and Distributed Environment setting).
- NFS (Network File System) performance tuning (file server tuning)

Performance monitoring using the WebSphere Commerce PMI module

You can monitor the performance of your WebSphere Commerce system by using the WebSphere Application Server Performance Monitoring Infrastructure (PMI). The function of the previous WebSphere Commerce performance monitor is now transferred to the WebSphere Commerce PMI module. Performance data can then be monitored and analyzed with a variety of tools available from WebSphere Application Server. Site Administrators can use the information gathered from the tools to detect performance problems and analyze performance trends. The tools can be used for measuring the performance of a WebSphere Commerce application server from a local or remote machine.

The WebSphere Commerce application server gathers statistics for URLs, tasks, and JSPs. Each data key has an associated set of counters that provide the following information:

- Average response time
- Last response time
- Minimum response time
- Maximum response time
- Hits
- Total response time
- Standard Deviation

To set up PMI, see the section "Monitoring performance" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

Setting up the PMI module

The Performance Monitoring Infrastructure (PMI) uses a client-server architecture. The server collects performance data from various WebSphere Application Server components. A client retrieves performance data from one or more servers and processes the data.

The WebSphere Application Server server collects PMI data in memory. This data consists of counters such as servlet response time and data connection pool usage. The data points are then retrieved using a Web client, Java client or JMX client. WebSphere Application Server provides the Tivoli® Performance Viewer, a Java client which displays and monitors performance data

In order to use the performance monitoring tools, you need to set up the WebSphere Commerce PMI module as follows:

1. In order to monitor performance data through the PMI interfaces, you must first enable the performance monitoring services through the WebSphere Application Server administrative console and restart the server. See the section "Enabling PMI services in application server through the administrative console" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>) for the detailed steps.
2. You can also monitor performance data through the PMI interface by enabling the performance monitoring service in NodeAgent through the administrative console. See the section "Enabling PMI services in NodeAgent through the administrative console" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>) for the detailed steps.
3. Once you have setup the above, you can collect the data.

The monitoring levels that determine which data counters are enabled can be set dynamically, without restarting the server. This can be done in one of three ways:

- Enable data collection through the WebSphere Application Server administrative console.
- Enable performance monitoring services through Tivoli Performance Viewer (formerly Resource Analyzer).
- Enable performance monitoring services using the command line

See the sections corresponding to the above tasks in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>) for the detailed steps.

4. Monitor and analyze performance data.

You can monitor and analyze data with several tools:

- a. Monitor performance data with Tivoli Performance Viewer. This tool is included with WebSphere Application Server.
- b. Monitor performance data with other Tivoli monitoring tools.
- c. Monitor performance data with user-developed monitoring tools. Write your own applications to monitor performance data.
- d. Monitor performance with third-party monitoring tools.

See the sections corresponding to the above tasks in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>) for the detailed information.

Starting up Tivoli Performance Viewer

The Tivoli Performance Viewer is a performance monitor for WebSphere Application Server shipped with WebSphere Application Server 5.0. The viewer retrieves performance data by periodically polling the administrative server. Data is collected continuously and retrieved as needed from within the viewer. Launch the Performance Viewer and specify the level of data to collect from the WebSphere Application Server Administration Console. Use the graphical interface to retrieve and view data in a table or chart, or to store data in a log file. You can

use the viewer not only use to monitor the Commerce PMI, but also other generic WebSphere Application Server performance data that are relevant to WebSphere Commerce.

For more information see the topic "Monitoring performance with Tivoli Performance Viewer (formerly Resource Analyzer)" in the the WebSphere Application Server Information Center

When you start the Tivoli performance viewer for WebSphere Commerce, you need to specify the SOAP or RMI port number for the `WC_instance_name` application server in order to measure the WebSphere Commerce related counters. Otherwise, by default, the Tivoli Performance Viewer will try to connect to the `server1` application server instead. One way to determine the `WC_instance_name` application servers's SOAP or RMI port number is to look into the `System.Out` log file of `WC_instance_name`. When the server starts up, the SOAP or RMI port number will be displayed.

Following are the steps to start the Tivoli Performance Viewer

1. Enable PMI services through the administrative console.
2. Start the Tivoli Performance Viewer. This can be done in two ways:
 - a. Start performance monitoring from the command line. Go to the `WAS_installdir/bin` directory and run the `tperfviewer` script.
On Windows 2000 environments, you can specify the host and port as:
`tperfviewer.bat host_name port_number connector_type`
On the AIX and other UNIX[®] platforms, use
`tperfviewer.sh host_name port_number connector_type`

For example: `tperfviewer.bat localhost 8879 SOAP`
The `connector_type` can be either SOAP or RMI. 8879 is the default ND port for a SOAP connector. 9809 is the default ND port for an RMI connector
 - b. Click **Start** → **Programs** → **IBM WebSphere** → **Application Server v.50** > **Tivoli Performance Viewer**.
The Tivoli Performance Viewer detects which package of WebSphere Application Server you are using and connects using the default Remote Method Invocation (RMI) connector port. If the connection fails, a dialog is displayed to provide new connection parameters. You can connect to a remote host or a different port number, by using the command line to start the performance viewer.
3. Adjust the data collection settings.
Refer to the instructions in the topic "Setting performance monitoring levels" in the WebSphere Application Server Information Center (<http://www.ibm.com/software/webservers/appserv/infocenter.html>).

WebSphere Commerce PMI Module report values

After you have started the Tivoli Performance Viewer, from the Resource Selection panel, expand **Commerce Counter Group**. The data will be grouped according to Tasks, URLs, and Views under the Store ID. Note that all the task commands are found only under `StoreId=0`. Once a particular URL, View or Task command is selected from the Resource Selection Panel, the Counter Selection Panel will display the WebSphere Commerce counter.

These are the counters that are included for each Task, URL, or view, note that all timing values are elapsed time:

Counter Name**Definition****Average response time**

Average response time of task

Last response time

The last response time of a task.

Minimum response time

Minimum response time of a task.

Maximum response time

Maximum response time of a task.

Hits The total number of times a task was called.

Total response time

Total response time of a task.

Standard deviation

Standard deviation of response time. The formula used to calculate the standard deviation assumes that the data conforms to a standard distribution.

Other performance tools

The Site Administrator may also need to use the following tools:

- IBM Tivoli Web Site Analyzer

By capturing, analyzing, storing and reporting on Web site usage, health, integrity and site content, the IBM Tivoli Web Site Analyzer can shed light on visitor site interactions and the site's overall performance. You can leverage this insight to optimize the site for increased customer loyalty and e-business effectiveness. Web Site Analyzer can track the popularity of page content and product purchases for targeted offers or for campaigns to specific visitors or customer segments. It can indicate where a decrease in investment or possible change in web navigation should occur due to less visited Web or product pages.

For more information about the Web Site Analyzer, refer to the following Web address:

<http://www.ibm.com/software/sysmgmt/products/web-site-analyzer.html>

- Commerce Studio Page Detailer

Use to analyze Web pages and display the identity, size, source, and time it takes to deliver each item on the page. You can use these details to identify areas where performance could be improved to enhance the end user experience. You can run Page Detailer without running the other components of Commerce Studio.

Maintaining the Scheduler

In order to ensure proper performance, the Scheduler must be adequately maintained. It is the Site Administrator's responsibility to make sure that the Scheduler is running properly by providing routine maintenance.

The SCHSTATUS table contains one entry for every execution of a job in the SCHCONFIG table. Because of the large volume of jobs run by the scheduler, job status records kept in the SCHSTATUS table become quite numerous. This can have an adverse affect on the performance of the WebSphere Commerce server. It is

therefore recommended that you periodically clean up the SCHSTATUS table using the Scheduler Status Display page. This is located in the Administration Console under the **Configuration** menu. Deleting job status records reduces the size of the SCHSTATUS table by cleaning up jobs based on time stamp and job reference number.

Alternatively, you can schedule the CleanJob job to run and automatically trim the size of the SCHSTATUS table. Use the endTime parameter to specify whether you want to remove job status records for the previous week or the previous month. For information on the syntax and scheduling of the CleanJob command, refer to the WebSphere Commerce Production and Development online help for your platform.

Note: When you run the CleanJob, the store publishing status is removed from the store publish page.

If you do not need to keep the history of the job, you can always set the autoClean property of the scheduler to "ON". This automatically removes the job from the SCHSTATUS table after the job finishes executing.

For more information on the Scheduler, see "Scheduling services" on page 21, or refer to the WebSphere Commerce online help.

Replication for LDAP

WebSphere Commerce supports using LDAP for authentication and for storing authentication and profile data. Some of the data is replicated between the WebSphere Commerce database and the LDAP server. Most of the replication can be configured using the ldapentry.xml file. Unlike previous versions of WebSphere Commerce where replication logic is triggered only from specific Member Subsystem commands, the current version of WebSphere Commerce performs replication at the object level which means replication occurs whenever it is necessary. As a result, if the LDAP server is down, an error will be generated.

Testing the site on a staging server

To test a site on a staging server, do the following:

1. Configure the staging server.
2. Create triggers for any custom tables.
3. Configure your remote database (if applicable).
4. Copy data to the staging database.
5. Test the site.
6. Run the Stage Check command to ensure there is no unique index key conflict.
7. Propagate data to the production database.
8. Copy the files to the production server.
9. Delete staged objects from the STAGLOG table.

Configuring the staging server

Any WebSphere Commerce machine can be set up as a staging server. A staging server can be configured during or after installation. Setting up a staging server during WebSphere Commerce installation is described in the *WebSphere Commerce Installation Guide*. To set up a staging server after installation, do the following:

1. Create and configure a separate WebSphere Commerce instance to use as a staging server with the Configuration Manager.
2. Ensure that you select the **Use Staging Server** check box on the **Database** panel to configure the instance as your staging server.
3. Ensure that caching is not enabled in the Configuration Manager **Cache** panel.

Creating triggers for custom tables

A trigger creates an entry in the STAGLOG table which identifies database record changes. You can modify the settings of the existing triggers to new tables if the tables contain the same data scope and key characteristics.

If you have not created any new tables, you do not need to perform this step. If you have created new tables, refer to the instructions in configuring the staging server for customized tables.

Configuring a remote database

If you have setup your staging server on a machine other than your production server, the remote database must be configured. If you plan to run staging utilities from the staging server, you need to configure your production database as the remote database in your staging server. If you plan to run staging utilities from the production server, you need to configure the staging database as the remote database in your production server.

 For a DB2 database, refer to the *DB2 Administration Guide*.

 For an Oracle database, refer to the product's documentation.

Copying data to the staging database

To copy data from the production database to the staging database, do the following:

1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written.
4. Type the following:

-  

```
stagingcopy -scope _all_ -sourcedb production_database_name
             -destdb staging_database_name
```

-  

```
stagingcopy -scope _all_ -sourcedb production_database_name
             -destdb staging_database_name -dbtype oracle -sourcedb_user
             user -sourcedb_passwd password -destdb_user
             user -destdb_passwd password
```

-     

```
. stagingcopy.sh -scope _all_ -sourcedb production_database_name
               -destdb staging_database_name -sourcedb_user user -destdb_user user
```

-   

```
. stagingcopy.sh -scope _all_ -sourcedb production_database_name
               -destdb staging_database_name dbtype oracle -sourcedb_user
               user -sourcedb_passwd password
               -destdb_user user -destdb_passwd password
```

Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

5. Examine the `stagingcopy_YYYY.MM.DD_HH.MM.SS.ZZZ.log` file to verify that the command was successful.
6. Stop and restart the staging server instance.

Running the Stage Check command

To check potential unique index key conflict between the staging database to the production database, do the following:

To run the Stage Check command to ensure there is no unique index key conflict, do the following:

1. Set the PATH environment variables.
2. Configure the database.
3. Change to the directory to which you want log files written.
4. Type the following:

-  

```
stagingcheck -scope _unique_index_ -sourcedb staging_database_name  
-destdb production_database_name
```

-  

```
stagingcheck -scope _unique_index_ -sourcedb staging_database_name  
-destdb production_database_name -dbtype oracle -sourcedb_user  
user -sourcedb_passwd password -destdb_user  
user -destdb_passwd password
```

-     

```
stagingcheck.sh -scope _unique_index_ -sourcedb staging_database_name  
-destdb production_database_name -sourcedb_user user -destdb_user user
```

-   

```
stagingcheck.sh -scope _unique_index_ -sourcedb staging_database_name  
-destdb production_database_name dbtype oracle -sourcedb_user  
user -sourcedb_passwd password -destdb_user  
user -destdb_passwd password
```

Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

5. Examine the `stagingcheck_YYYY.MM.DD_HH.MM.SS.ZZZ.log` file to verify that the command was successful.

Propagating data to the production database

To propagate data from the staging database to the production database, do the following:







1. Set the PATH environment variable.
2. Configure the database.
3. Change to the directory to which you want log files written.
4. Type the following:




-  

```
stagingprop -scope _all_ -sourcedb staging_database_name -destdb  
production_database_name
```

-  

```
stagingprop -scope _all_ -sourcedb staging_database_name
            -destdb production_database_name -dbtype oracle -sourcedb_user
            user -sourcedb_passwd password -destdb_user
            user -destdb_passwd password
```

-      

```
. stagingprop.sh -scope _all_ -sourcedb staging_database_name
  -destdb production_database_name -sourcedb_user user -destdb_user user
```
-   

```
. stagingprop.sh -scope _all_ -sourcedb staging_database_name
  -destdb production_database_name dbtype oracle -sourcedb_user
  user -sourcedb_passwd password -destdb_user
  user -destdb_passwd password
```

Note: Use *host:port:sid* for the Oracle database name. For example, *myhost:1521:mydb*.

5. Examine the `stagingprop_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For further information on propagating data to the production database, see the examples.

Copying files to the production server

If you add, change, or delete image or HTML files (as opposed to data in the database) on the staging server, you must manually copy these files to the production server.



To copy files to the production server, do the following:



1. Copy the files to the production server in the correct directory:
 - a. Create a zipped file that contains all static HTML files, associated image files, and other embedded files. This file should contain new, updated, and unchanged files.
 - b. Transfer the zipped files to the production server.
 - c. Unzip each zipped file into the corresponding directory in the production server directory structure. If you have moved the HTML files, edit the WebSphere Commerce configuration to point to the files in the new directory.
2. Delete unused directories on the production server.






Deleting staged objects

To delete staged objects, do the following:

1. Set the PATH environment variable.
2. Change to the directory to which you want log files written.
3. Type the following:

-  

```
dbclean -object staglog -type obsolete -db dbname
        -days daysold -loglevel loglevel
```
-  

```
dbclean -object staglog -type obsolete -db dbname
        -days daysold -loglevel loglevel -dbtype oracle
        -dbuser user -dbpasswd password
```
-     

```
. dbclean.sh -object staglog -type obsolete -db dbname
-dbuser user -days daysold -loglevel loglevel
```

•   

```
. dbclean.sh -object staglog -type obsolete -db dbname
-days daysold -loglevel loglevel -dbtype oracle
-dbuser user -dbpasswd password
```

Note: Use *host:port:sid* for the Oracle database name. For example,
myhost:1521:mydb.

4. Examine the `dbclean_yyyy.mm.dd_hh.mm.ss.zzz.log` file to verify that the command was successful.

For additional examples on deleting staged objects, refer to examples of deleting objects.

WebSphere Commerce Payments performance tuning parameters

WebSphere Commerce Payments contains tuning parameters that allow you to control WebSphere Commerce Payments internal resource allocation.

Attention: These parameters should only be modified by an administrator who is very familiar with WebSphere Commerce Payments. Setting either of these parameter values too high could cause performance degradation or an outright failure of WebSphere Commerce Payments at startup time.


It is *highly* recommended that you:

- Make changes in small increments only.
- Change only one parameter at a time; then measure the effect before making another change.
- Thoroughly test changes on a non-critical system before using them on a production system.

WebSphere Commerce Payments manages the following sets of thread pools:

Protocol thread pool

The threads in this pool are assigned the task of processing *server-side* protocol-specific messages within a cassette. There is one such pool for each cassette that defines its own ComPoints.

 The property, `wpm.ppoolsize`, changes the protocol thread pool size.

The pool size can be changed by completing the following steps in the WebSphere Application Server Administrative Console

1. Expand **Servers**.
2. Click **Application Servers>wpm Commerce Payments Server>Process Definition>Java Virtual Machine>Custom Properties>wpm.ppoolsize**.
3. Type the desired **Value**. The default is 8.
4. Click **Apply**.


.

Given the nature of the processing that is typically involved with server-side protocol messages (increased levels of network communication), these threads are typically held for a long time for each given request.

Therefore, a cassette that relies on protocol threads to process protocol messages should be configured with a large enough protocol thread pool to accommodate a typical number of concurrent transactions involving the significant protocol messages.

Service thread pool

The threads in this pool are used by cassettes as well as the framework to perform current or future tasks in the background. The default number of threads in this pool is 6.

 The property `wpm.spoolsize` changes the Service thread pool size.

You can change the pool size by doing the following in the WebSphere Application Server Administrative Console.

1. Expand **Servers**.
2. Click **Application Servers>wpm Commerce Payments Server>Process Definition>Java Virtual Machine>Custom Properties>wpm.spoolsize**.
3. In the **Value** field, specify the desired pool size. The default is 6.
4. Click **Apply**.

Initially, the most common uses for service threads were to process tasks that were scheduled to execute at some point in the future. Examples of such tasks are retried transactions with the cassette's back end processor, as well as periodic maintenance tasks.

`wpm.disableDuplicateOrderCheck`

This parameter instructs WebSphere Commerce Payments to bypass duplicate order checking when processing a new order and eliminate access to the database. This results in higher transaction throughput. It is recommended to use this parameter only if your merchants will not be generating duplicate order numbers. Doing so causes these orders to fail. To enable this parameter, complete the following steps through the WebSphere Application Server Administrative Console:

1. Expand **Servers**.
2. Click **Application Servers>wpm Commerce Payments Server>Process Definition>Java Virtual Machine>Custom Properties**.
3. Click **New**.
4. In the **Name** field, type the parameter name `wpm.spoolsize`.
5. Type **1** for the **Value**.
6. Click **Apply**.

Tip: Your database product may limit the number of concurrent connections that you can establish to your database at a given time. This may either be a technical limit or one imposed by the license agreement under which you purchased the database product. These factors must be considered while adjusting the above parameters. If you exceed such a limit by setting either of these parameters too high, the Payment Servlet initialization will fail.

Managing the Payments instance database on iSeries

As new orders and payments are processed by WebSphere Commerce Payments, the Payments instance database collection will increase. If the iSeries storage space begins to fill up, the system operator may receive warning messages such as CPI099C - Critical storage lower limit reached. If that happens you should consider freeing up space in the WebSphere Commerce Payments instance database.

The WebSphere Commerce Payments instance database collection is created with journaling set to *YES. This means that the instance library will contain database journal receivers (entries that indicate when events such as changes to a database file occur). In iSeries, the journal QSQJRN is built in the WebSphere Commerce Payments instance library, and journal receivers named QSQJRNXXXX are created in the instance library. Old journal receivers in the instance library (named QSJRNXXXX with type *JRNRCV) may be deleted to free up space if the WebSphere Commerce Payments instance library is getting too large. The most recent journal receiver will still be attached to the database and should not be deleted. In addition, the database tables may be pruned by using the Commerce Payments Pruneorders utility available from IBM service.

For more information about cleaning up storage, refer to the *AS/400 System Operation* manual (SC41-4203).

Chapter 8. Troubleshooting

Site Administrators are responsible for troubleshooting and finding solutions to system problems. This section covers troubleshooting for the Adapter for Crossworlds[®], and WebSphere Commerce Payments. Refer to the *WebSphere Commerce Installation Guide* for information on troubleshooting the following:

- Downloadable tools
- Log files
- WebSphere Application Server problems
- Web Server problems
- WebSphere Commerce problems
- Database problems

Adapter for CrossWorlds[®]

This section lists potential problems you may encounter while configuring the adapter, and actions to resolve these problems.

- **Problem:** You have the InterChange Server (ICS) and repository on one machine, and the WebSphere Commerce Server on another machine. What configuration information is needed for the Visigenic ORB agent to work across subnets? The symptom is a dialog box that pops up when attempting to connect, indicating that the WebSphere Commerce Server "Can't locate the InterChange Server '<server name>'. The server is probably not running.", even though you can independently verify that the server is running.

Solution 1:

1. Create a user-level environment variable, named VBROKER_ADM which points at the adm directory in your Visibroker installation (i.e. d:\inprise\vbroker\adm).
2. In the adm directory, create a text file named agentaddr which contains the IP addresses or hostname(s) of the client machine which you need to connect to this ICS.
3. Restart the osagent process, making sure that VBROKER_ADM is in the environment of the user or shell that starts this process. You should now be able to connect your WebSphere Commerce Server to the ICS. Note that each osagent must have its own agentaddr text file that contains the remote IP of the WebSphere Commerce Server.

Note: There is no extension on agentaddr, and the only osagent that should be running is on the ICS side.

Solution 2: Others have had better success with an alternate environment variable, OSAGENT_ADDR_FILE, which should specify the path to and filename of the agentaddr file.

Because the ORB needs a few minutes of discovery time, attempt reconnecting over a ten minute window before giving up on any particular method.

- **Problem:** The WebSphere Commerce Server is unable to connect to more than one InterChange Server. The OSAGENT_ADDR environment variable was set on the WebSphere Commerce Server machine but can only connect to the InterChange Server running on the machine specified. You are running two InterChange Servers on two different subnets, and a WebSphere Commerce Server on a third

machine. An osagent is running on each InterChange Server machine and each machine has the VBROKER_ADM environment variable configured properly to point to the directory containing the agentaddr file which contains the IP address of each InterChange Server/osagent machines.

Solution: Create a localaddr text file in the directory specified in the VBROKER_ADM environment variable of each osagent machine, and specify the IP address or the DNS alias of the other osagent machine.

- **Problem:** You are trying to connect an NT machine's WebSphere Commerce Server to an ICS running on Solaris.

Solution: Make sure that the agentaddr file for both the WebSphere Commerce Server and ICS locations have not only the IP address of the machine being connected to, but the machines own IP address as well. Therefore, the agentaddr file on the NT machine must have both it's own IP address and the IP address of the Solaris machine.

Notes:

1. For other problems not addressed in this section, contact your IBM support representative.
2. The Adapter for CrossWorlds[®] is not supported on iSeries, Linux, IBM eServer[™] zSeries[®], or S/390[®] Linux.

Note: For other problems not addressed in this section, contact your IBM support representative.

WebSphere Commerce Payments

Each business model possesses unique requirements and, as such, utilizes the WebSphere Commerce Payments functions somewhat differently. Often, WebSphere Commerce Payments can be tuned to maximize its performance and functionality for a given business environment.

Tuning for high-performance environments

Wait time for transmission control protocol/internet protocol (TCP/IP) sockets

Each request to WebSphere Commerce Payments results in a TCP socket going into the TIME_WAIT state and remaining there for several minutes. For machines that service a high volume of requests, there may be a large number of sockets in the TIME_WAIT or TIME_CLOSED state, resulting in rejected requests, that is, a return code of Cannot connect to WebSphere Commerce Payments. This behavior is expected and necessary for all TCP connections.

TCP sockets move into the TIME_WAIT state for a period of time to ensure that any subsequent communication on the socket is not mistaken for new communication on a newly-bound socket. This period of time is, theoretically, 2 MSL (that is, twice the maximum segment lifetime). In practice, the TIME_WAIT default is four minutes on the Windows operating system and Solaris and two minutes on AIX. The default is approximately two minutes on iSeries systems. By altering the TIME_WAIT values on your operating system, high-volume users can reduce this problem. Following are examples of how the interval can be reconfigured on the Windows, AIX and Solaris operating systems.

Exception: If an alternative stack is being used, other measures may be required.

Windows:

1. Locate in the Registry:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\tcpip\Parameters  
\TcpTimeWaitDelay
```

If this entry does not exist in your Windows Registry, you must create it, following step 2 below.

2. Edit this entry as a new DWORD item.
3. Set it to any value between 30 and 300 (value represents the number of seconds). It is recommended that you set this value to 30.

AIX:

```
no -o tcp_timewait=1
```

The value (1) is in 15 second increments. For example, 1 equates to 15 seconds, 2 equates to 30 seconds, and so on. It is recommended that you set the value to 1 or 2.

Solaris:

```
ndd -set /dev/tcp tcp_close_wait_interval 30000
```

The value (30000) is in milliseconds.

Appendix A. LDAP scenario

LDAP scenario: the LDAP server as the member repository

One scenario for using the LDAP server as a member repository is that you create a new instance of WebSphere Commerce and specify it to use an LDAP server as the member repository. In this scenario, the WebSphere Commerce database is populated with bootstrap data for the Root Organization, Default Organization, and the 'wcsadmin' user entry.

For this scenario, you are required to complete the following:

1. Create suffixes that are required in the directory server. User and organizational entity entries that will be used by WebSphere Commerce will exist under these suffixes.
2. Create entries on the directory server for the 'Root Organization', 'Default Organization', and 'wcsadmin', with both 'Default Organization' and 'wcsadmin' under the 'Root Organization'. Refer to the "Creating bootstrap entries in the LDAP server" related link below for a sample LDIF file.
3. Create a password in the LDAP server for the 'wcsadmin' user entry.
4. After the entries are created in the directory server, you should update the DN column in the ORGENTITY table with the correct distinguished names for the Root Organization (with MEMBER_ID -2001) and the Default Organization (with MEMBER_ID -2000).
5. Set up the `ldapentry.xml` file for mapping WebSphere Commerce attributes to LDAP attributes. Ensure search bases for users are specified in the `ldapentry.xml` file such that the 'wcsadmin' user can be found.
6. You log on as 'wcsadmin' providing the proper password.

As more users are created on the directory server through WebSphere Commerce or another application, the users can log on WebSphere Commerce using either their RDN value or DN value. If users will logon using RDN, ensure search bases are specified properly in the `ldapentry.xml` file.

Note: If you create multiple instances of WebSphere Commerce and they share the same LDAP server, you should ensure the configuration for the LDAP server within each WebSphere Commerce instance is the same.

Appendix B. LDAP files

This appendix gives a detailed description of the following LDAP files:

- ldapmap.dtd
- ldapentry.xml

ldapmap.dtd

ldapentry.dtd replaces the ldapmap.dtd file used in the previous version of WebSphere Commerce. The following is the DTD used for LDAP. You should not modify this DTD.

```
<!-- new mapping file -->
<!ELEMENT ldapentry (entry+)>
<!ELEMENT entry (ldapsetting,ldapmap)>
<!ELEMENT ldapsetting (ldaprtn,ldapocs,ldapbase)>
<!ELEMENT ldaprtn EMPTY>
<!ELEMENT ldapocs EMPTY>
<!ELEMENT ldapbase EMPTY>
<!ELEMENT ldapmap (map+)>
<!ELEMENT map (objectAttribute+,objectSeparator?,ldapAttribute)>
<!ELEMENT objectAttribute EMPTY>
<!ELEMENT objectSeparator EMPTY>
<!ELEMENT ldapAttribute EMPTY>

<!-- ATTLLIST entry
entryName      (User|Organization|OrganizationalUnit) #REQUIRED>

<!-- ATTLLIST ldapbase
defaultBase    CDATA      #REQUIRED
searchBase     CDATA      #REQUIRED>

<!-- ATTLLIST ldaprtn
rdnName        CDATA      #REQUIRED  keyAttrName    CDATA      #REQUIRED
keyObjName     CDATA      #REQUIRED>

<!-- ATTLLIST ldapocs
objClass       CDATA      #REQUIRED>

<!-- ATTLLIST objectAttribute
attrName       CDATA      #REQUIRED>

<!-- ATTLLIST objectSeparator
attrSeparator  CDATA      #IMPLIED>
<!-- ATTLLIST ldapAttribute
name           CDATA      #REQUIRED
operation      (replace|add) #REQUIRED
flow           (ldapToWcs|wcsToLdap|bothDirections) #REQUIRED>
<!-- End of DTD -->
```

ldapentry.xml

The ldapentry.xml file replaces the ldapmap.xml file used in the previous version of WebSphere Commerce. Ensure contents of your ldapmap.xml file used previously are transferred to the ldapentry.xml file. The ldapentry.xml file provides a default mapping for the most commonly used fields. LDAP attributes in the default schema mapping are based on the default user objectclasses (person,

organizationalPerson, inetOrgPerson and ePerson) and default organizational entity object classes (organization and organizationalUnit). These object classes are defined for all LDAP servers supported by WebSphere Commerce. To extend the default schema mapping beyond these object classes, ensure that schema extensions are performed on the LDAP server first, then mapping for new attributes can be placed in the ldapentry.xml file. It is recommended that you do not persist the following attributes to LDAP: internally generated keys within WebSphere Commerce, data that changes often, such as the Lastsession in the USERS table.

For information about LDAP attributes refer to the following site:
www.as400.ibm.com/ldap/schema.

```
<?xml version="1.0"?>
<!DOCTYPE ldapentry SYSTEM "ldapentry.dtd">
<ldapentry>
  <entry entryName="User">
    <ldapsetting>

<ldaprdn rdnName="uid"
  keyAttrName="logonId" keyObjName="UserRegistry"/>
    <ldapocs
objClass="top;person;organizationalPerson;inetOrgPerson"/>
    <ldapbase
defaultBase="o=Default Organization,o=Root Organization"
  searchBase="o=Root Organization"/>
    </ldapsetting>
    <ldapmap>
      <map>

        <
objectAttribute attrName="logonPassword"/>

<ldapAttribute name="userPassword"
  operation="replace" flow="wcsToLdap"/>
      </map>
      <map>

<objectAttribute attrName="lastName"/>

<objectAttribute attrName="firstName"/>

<objectSeparator attrSeparator="/" />

<ldapAttribute name="cn" operation="replace"
  flow="wcsToLdap"/>

    </map>
    <map>

      <objectAttribute attrName="lastName"/>

      <ldapAttribute name="sn" operation="replace"
" flow="bothDirections"/>

    </map>
    <map>
```

```

<objectAttribute attrName="firstName"/>

<ldapAttribute name="givenName" operation="replace
" flow="bothDirections"/>
  </map>
  <map>

<objectAttribute attrName="phone1"/>

<ldapAttribute name="homePhone" operation="replace
" flow="bothDirections"/>
  </map>
  <map>

<objectAttribute attrName="zipCode"/>

<ldapAttribute name="postalCode" operation="replace
" flow="bothDirections"/>
  </map>
  <map>

<objectAttribute attrName="address1"/>

<objectAttribute attrName="address2"/>

<objectAttribute attrName="address3"/>

<objectSeparator attrSeparator="/"/>

<ldapAttribute name="postalAddress" operation="replace
" flow="bothDirections"/>
  </map>
  </ldapmap>
</entry>

entry entryName="Organization">
  <ldapsetting>

  <ldaprdn rdnName="o" keyAttrName="orgEntityName"
keyObjName="Organization"/>
  <
  ldapocs objClass="top;organization"/>
  <
  ldapbase defaultBase="o=Root Organization" searchBase="
o=Root Organization"/>
  </ldapsetting>
  <ldapmap>
  <map>

  <objectAttribute attrName="businessCategory"/>

  <ldapAttribute name="businessCategory" operation="
replace" flow="bothDirections"/>
  </map>
  <map>

  <objectAttribute attrName="description"/>

  <ldapAttribute name="description" operation="
replace" flow="bothDirections"/>
  </map>
  <map>

  <objectAttribute attrName="address1"/>

  <objectAttribute attrName="address2"/>

```

```

<objectAttribute attrName="address3"/>

<objectSeparator attrSeparator="/"/>

<ldapAttribute name="postalAddress" operation="replace
" flow="bothDirections"/>
    </map>
</map>

<objectAttribute attrName="phone1"/>

<ldapAttribute name="telephoneNumber" operation="replace
" flow="bothDirections"/>
    </map>
</ldapmap>
</entry>
<entry entryName=
"OrganizationalUnit">
    <ldapsetting>

        <ldaprdn rdnName="ou" keyAttrName="orgEntityName"
keyObjName="Organization"/>

        <ldapocs objClass="top;organizationalUnit"/>

        <ldapbase defaultBase="o=Root Organization"
searchBase="o=Root Organization"/>
            </ldapsetting>

            <ldapmap>
                <map>

                    <objectAttribute attrName="businessCategory"/>

                    <ldapAttribute name="businessCategory" operation="replace
" flow="bothDirections"/>
                        </map>
                    </map>

                    <objectAttribute attrName="description"/>

                    <ldapAttribute name="description" operation="replac
e" flow="bothDirections"/>
                        </map>
                    </map>

                    <objectAttribute attrName="address1"/>

                    <objectAttribute attrName="address2"/>

                    <objectAttribute attrName="address3"/>

                    <objectSeparator attrSeparator="/"/>

                    <ldapAttribute name="postalAddress" operation="replace
" flow="bothDirections"/>
                        </map>
                    </map>

                    <objectAttribute attrName="phone1"/>

                    <ldapAttribute name="telephoneNumber" operation="replace
" flow="bothDirections"/>

```

```

        </map>
    </ldapmap>
</entry>
</ldapentry>

```

The format of the `ldapentry.xml` file is as follows:

entry The entry element identifies which type of member the mappings are for. Valid values are `User`, `Organization`, or `OrganizationalUnit`.

ldapsetting - ldaprdn - rdnName

Specifies which LDAP attribute is the RDN attribute

ldapsetting - ldaprdn - keyAttrName

Specifies which WebSphere Commerce attribute maps to the RDN attribute. For users, the attribute name corresponds to the property name documented in the syntax of the `UserRegistrationAdd` command. The attribute name is case sensitive. Similarly for organization and organizational unit in which case the `OrgEntityAdd` command should be used.

ldapsetting - ldaprdn - keyObjName

Specifies which WebSphere Commerce access bean can the WebSphere Commerce attribute specified in `keyAttrName` be found

ldapsetting - ldapocs - objClass

Specifies the LDAP object classes that are used to create LDAP entries

ldapsetting - ldapbase - defaultBase

Specifies the first search base DN under which WebSphere Commerce will search for users during logon if RDN is used for logon. For organizational entities, `defaultBase` specifies the first search base DN under which WebSphere Commerce will search for organizational entities whenever it requires.

ldapsetting - ldapbase - searchBase

Specifies the rest of the base DN's under which WebSphere Commerce will search for users and organizational entities.

map -objectAttribute - attrName

A WebSphere Commerce attribute name. For users, the attribute name corresponds to the property name documented in the syntax of the `UserRegistrationAdd` command. The attribute name is case sensitive. Similarly for organization and organizational unit in which case the `OrgEntityAdd` command should be used.

map -ldapAttribute - name

Name of an LDAP attribute to be mapped to the WebSphere Commerce attribute specified in `attrName`

map -ldapAttribute - flow

Specifies whether the attribute value is read from, written to, or both read and written to LDAP. Valid values are `ldapToWcs`, `wcsToLdap`, or `bothDirections`.

map -ldapAttribute - operation

Specifies whether how the attribute value should be modified for LDAP. Valid values are `replace` or `add`. A value of `replace` updates the current user information with the new information provided (for example, replace an existing phone number with a new one). A value of `add` includes another entry for the user (for example, adding a new phone number to the current user information, making a list of phone numbers for the user).

map - objectSeparator - attrSeparator

Separator character used when storing or retrieving multiple WebSphere Commerce attributes to or from a single LDAP attribute.

Appendix C. WebSphere Commerce Payments Tutorial

This tutorial guides you through a first-time WebSphere Commerce Payments setup with the OfflineCard Cassette. As part of this initial setup, and to demonstrate the most common administration and payment functions, WebSphere Commerce Payments provides tutorial support using the OfflineCard Cassette and a Sample Checkout. For detailed information on administration, configuration and payment functions, see the online help for the WebSphere Commerce Payments user interface.

Following are the six must-do configuration tasks to set up an operational WebSphere Commerce Payments with the OfflineCard Cassette:

1. Access the WebSphere Commerce Payments user interface
2. Create a WebSphere Commerce Payments merchant and authorize a payment cassette
3. Define WebSphere Commerce Payments users
4. Assign user roles
5. Create an account
6. Create a brand

Once the configuration tasks are complete, you are ready to place an order and begin the following payment processing tasks that merchants typically perform on a daily basis:

7. Create an order
8. Approve orders
9. Deposit payments
10. Settle batches
11. Issue credits
12. View daily batch totals

Step 1: Accessing the WebSphere Commerce Payments user interface

The first step is to log on to Payments as the Payments Administrator.

To log on to the WebSphere Commerce Payments user interface, do the following:

1. In a Web browser point to `http://host_name:port/webapp/PaymentManager` where *host_name* is the host name running the Web Server for Payments, and *port* refers to the port number Payments is running on.
2. Type your WebSphere Commerce Site Administrator ID.
3. Type your WebSphere Commerce Site Administrator password.
4. Click **Logon**.

Important: If the HTTP server that the WebSphere Commerce Payments instance is using is configured for a port number other than the default, include the port number following the host name in the WebSphere Commerce Payments Web address links throughout this tutorial.

The icons in the upper right of the user interface have the following uses:

- Click the multidirectional arrow to refresh the page.

- Click the left-pointing arrow to return to the last page visited.
- Click the question mark to access context sensitive online help for the page.

Step 2: Creating a WebSphere Commerce Payments merchant and authorizing a cassette

If you have not already done so, log on to WebSphere Commerce Payments as the Payments Administrator. You now have global views and global authority. The first step in configuring WebSphere Commerce Payments is to create a merchant and authorize that merchant to use a payment cassette. Do the following to create a merchant and authorize a cassette:

1. From the navigation frame click **Merchant Settings**.
2. From the Merchant Settings page click **Add a Merchant**.
3. On the Merchant Settings page, type the following information (note that the text in monospace must be entered in these fields for the tutorial):

Table 11. Create a merchant fields

Field name	Description
Merchant name	Type Test Store. This is the name that you assign to the merchant. Its only function is to display in the user interface.
Merchant number	Type 123456789. This is a number that you assign which uniquely identifies the merchant in all transaction data.
Authorized cassettes	Check the box for the OfflineCard . Checking this box authorizes the merchant to use this payment cassette.

4. Click **Create Merchant** to save the merchant configuration.

If you have already created a merchant whom you want to authorize this cassette to use, do the following:

1. Click **Merchant Settings**.
2. Click the Merchant Name.
3. Select the box for **OfflineCard**.
4. Click **Create Merchant**.

The merchant is now authorized to use this cassette.

Step 3: Defining WebSphere Commerce Payments users

For this tutorial, you will work with the following users:

- The default user created during installation (For more information see the *WebSphere Commerce Installation Guide*.)
- *Pat*, a user you will define

Use the WebSphere Commerce Organization Administration Console to accomplish tasks such as defining and managing users. Defining users in WebSphere Commerce Payments is a two-part process. For example, to define the user, *Pat*, you must use the WebSphere Commerce Organization Administration Console and assign Pat the role of Site Administrator. Then, you can assign Pat's user role to Merchant Administrator within the Payments UI directly or through the Organization Administration Console. Note that before you can assign access to a user, you must create a merchant.

To configure Payments users, do the following:

1. In a Web browser point to `https://host_name:port/adminconsole`
2. Click **Access Management>Users**.
3. Click **New**.
4. Create the new user, Pat, using the New User wizard.
5. From the Access page, assign Pat the role of Site Administrator.
6. Click **Payments>Users**.
7. Type the **User name** Pat.
8. Click **Search**.
9. Click Pat's name. The User's configuration page displays.
10. Select the desired merchant name.
11. Select the radio button for the desired role, in this case, Merchant Administrator.
12. Click **Update** to save your specifications.

If you are installing WebSphere Commerce Payments on a system remotely from WebSphere Commerce products and would like to use the WebSphere Commerce stylesheet in your WebSphere Commerce Payments user interface, you must copy the `PMCustomUI.properties` file in the `Payments_installdir\samples\wcs\PMCustomUI.properties` directory to the main WebSphere Commerce Payments installation directory.

Step 4: Assigning user roles

Users must be assigned to one of the following WebSphere Commerce Payments roles, which have relative mappings to the corresponding WebSphere Commerce roles:

Table 12. Role mapping

Payments role	WebSphere Commerce role
Payments Administrator	Site Administrator
Merchant Administrator	Site Administrator
Supervisor	Operations or Sales Manager
Clerk	Customer Service Supervisor

After creating the following users:

- A user, *Pat*
- A merchant, *Test Store*

you are ready to assign Pat's role in the WebSphere Commerce Payments configuration.

Exception: You can also assign the role of *No WebSphere Commerce Payments access* to deny users access to WebSphere Commerce Payments. This may be useful if you have a temporary need to deny a user access, for example, an employee on leave of absence. For more information on WebSphere Commerce Payments role permissions, see the Role Permissions Table in the *WebSphere Commerce Payments Programming Guide and Reference*.

To assign Pat the role of Merchant Administrator for the Test Store, in the Payments UI do the following:

1. From the navigation frame click **Users**.
2. On the Users Search page, type the user name Pat and click **Search**.
3. From the Users page, click the user name **Pat**.
4. From the **Merchant** scroll box, select **Test Store**.
5. Select the radio button for **Merchant Administrator**.
6. Click **Update** to save the user configuration.

At this point, log off the WebSphere Commerce Payments user interface and log on again, this time as the Merchant Administrator, Pat.

Important: In hosted environments where a Commerce Service Provider (CSP) establishes a WebSphere Commerce Payments that remotely services multiple merchants, the CSP is the Payments Administrator and acts per each merchant's contract with the provider. In this scenario the merchant configures their own merchant settings with Merchant Administrator authority granted by the CSP (Payments Administrator).

Logging on as the Merchant Administrator

To log off and log on again, do the following:

1. From the navigation frame, click **Logoff admin** to return to the main WebSphere Commerce Payments Logon page.
2. Type the **User ID** Pat.
3. Type the **Password** defined for Pat.
4. Click **OK**.

For the remainder of this tutorial, your role will be the user Pat, with Merchant Administrator authority for the Test Store. Your view of the WebSphere Commerce Payments user interface is limited to merchant administration functions; where, as the Payments Administrator, you had a global view of both Merchant and Payments Administrator functions.

Step 5: Creating an account

So far, you have defined one merchant, the Test Store, and enabled one payment cassette, the OfflineCard Cassette. Your first task as the Merchant Administrator is to establish an *account* for the OfflineCard Cassette.

An account is a relationship between the merchant and the financial institution which processes transactions for that merchant. There can be multiple accounts for each payment cassette. For this tutorial, you will create one account for the OfflineCard Cassette.

To create an account, do the following:

1. From the navigation frame click **Merchant Settings**.
2. From the Merchant Settings page, click the OfflineCard Cassette icon for the Test Store.
3. From the OfflineCard Cassette page, click **Accounts**.
4. On the Accounts page, click **Add an Account**.
5. Complete the following fields (note that the text in monospace must be entered for the tutorial):

Table 13. Add an account fields

Field name	Description
Account name	Type Test Account. This is the name that you assign to the account. Its only function is to display in the user interface.
Account number	Type 11111111. This is a number that either the hosting service provider or the Merchant Administrator, assigns to uniquely identify the account in all transaction data.
Financial Institution name	Type Test Bank. This is the name of the financial institution with which you hold this account. Its only function is to provide display information in the user interface.
Currency	Select the currency to be collected by this account.
Batch close time	The number of minutes past midnight that the cassette automatically closes batches for this account. A value of 0 (zero) represents midnight. 1439 is the maximum value allowed. A null value disables automatic batch closing.

6. Click **Create account** to create the account for the OfflineCard Cassette.

Specifying account settings

To use the tutorial, it is not necessary to specify any particular basic or advanced settings for the account and brand you just created for a cassette. You should be aware, however, that you can specify certain options for the account. For example, as Merchant administrator, you can define whether the account should have an expiration period associated with payment approvals.

To view account settings from the WebSphere Commerce Payments user interface for the OfflineCard Cassette, do the following:

1. Select **Merchant Settings** on the navigation frame.
2. From the Merchant Settings window, click the OfflineCard Cassette icon in the Test Store window.
3. From the OfflineCard Cassette window, click **Accounts**.
4. Select **Test Account** on the Accounts window.
5. From the Test Account window, select **Account Settings**.
6. You can view or change basic or advanced settings by clicking on the appropriate selection. For example, **select Advanced** settings to see the Approval Expiration setting for the Automated Payment Processing Option.
7. Enter options as appropriate and select **Update** to update the account settings. (To cancel, select the Back arrow on the Account Settings window.)

You can refer to the online help for more information about settings for the processing options. For now, however, we will explore one of them—the Approval Expiration option.

You may want to set an approval expiration to control the length of time a payment approval is valid. It may be useful to set an approval expiration period to avoid charges imposed by your financial institution for depositing funds after an approval expires. For example, for credit card orders, payment approval lasts for only a certain number of days as established by the credit card issuer. If you try to deposit funds for an order payment after the approval has expired, your financial institution may either reject the funds deposit or charge you more to make the

deposit. By setting an approval expiration period in WebSphere Commerce Payments you can avoid this situation because WebSphere Commerce Payments will prevent the capturing of funds for which an approval authorization has expired.

If a payment approval has expired and you try to deposit funds you will receive a message indicating the payment is not in the approve state. In the WebSphere Commerce Payments user interface, you can void the payment with an amount of 0. Once the payment is voided, you can perform an Approve operation to place the payment back into the approve state.

To use the approval expiration option, the cassette you use must support approval expiration. The OfflineCard, VisaNet, Paymentech and CustomOffline cassettes provided with WebSphere Commerce Payments support approval expiration.

Important: The value you enter as the approval expiration period applies to all brands associated with an account. If your brands each have different payment approval expiration rules, set the approval expiration value for the lowest common denominator, or create a separate account for each brand. If you create a separate account for each brand, you must settle batch payments separately for each account.

More information about approving orders, depositing payments, and settling batches is provided later in this chapter.

Step 6: Creating a brand

To use the OfflineCard Cassette, you must first configure the brand names of the credit card companies your store works with. This enables the shopper to select the appropriate credit card from a list during checkout. To create a brand for the OfflineCard Cassette, do the following (note that for this tutorial, ROBO is used for the brand):

1. From the navigation frame click **Merchant Settings**.
2. From the Merchant Settings page, click the OfflineCard Cassette icon for the Test Store.
3. From the OfflineCard Cassette page, click **Accounts**.
4. From the Accounts page, click the account name, Offline Account.
5. From the Offline Account page, click **Brands**.
6. From the Brands page, click **Add a Brand**.
7. In the **Brand Name** field, type ROBO.
8. Click **Create Brand**.

Step 7: Creating orders using the Sample Checkout

As the Merchant Administrator, you have global merchant authority, which means that you can do the following:

- Merchant-specific administration functions
- All payment processing functions

In a real business scenario, you may delegate payment processing tasks to other merchant-defined users who possess limited payment processing authorities (such as, Supervisor and Clerk). In this tutorial you as the Merchant Administrator will

perform these tasks. Having completed all of WebSphere Commerce Payments and merchant administration tasks necessary to begin payment processing, you are now ready to start:

- Approving orders
- Depositing payments
- Settling batches
- Issuing credits
- Viewing daily batch totals

For this tutorial you will use the Sample Checkout tool to create three orders for payment processing. The Sample Checkout tool provides a user interface with which you can create sample orders to test your cassette implementation. To access Sample Checkout, you must first change the default user ID and password as described below.

To access the WebSphere Commerce Payments Sample Checkout and create orders, do the following:

1. From the directory:

```
was_installdir/installedApps/host_name/  
payments_instance_Commerce_Payments_App.ear/SampleCheckout.war
```

open the configuration file `SampleCheckout.xml`.

2. At the `SampleCheckout` element, change the following attribute values:

```
pmHostName="fully_qualified_host_name "  
pmPort="port "  
default_userid="wc_userid "  
password="wc_password "
```

3. Save the file. Point your browser to `http://host_name:port/webapp/SampleCheckout`, where *host_name* is the host name of the machine running the Web Server for Payments, and *port* refers to the port number Payments is running on.
4. At the Sample Checkout page enter the following information (note that the monospaced text must be entered in these fields for the tutorial):

Table 14. Sample Checkout fields

Name	Description
Merchant number	Type the OfflineCard Sample Checkout merchant number, 987654321.
Order number	Type any number to represent an order number.
Amount	Type any amount to represent the total numeric amount of the order.
Currency	Type US dollar. The currency used to place this order.
Payment method	Select OfflineCard as the payment type.
Brand	Select the brand of OfflineCard you wish to use.
Credit card number	Type 4111111111111111.
Expiration Date	Select any future expiration month and year for your credit card.

▶ 400 For an iSeries system you must specify the brand of the credit card that you are using.

5. Click **Buy**.

Repeat these steps two more times so that you have three orders for which to process payments.

Step 8: Approving orders

Once you have created three orders using the Sample Checkout, you can approve those orders. Follow these steps to approve an order:

1. Point your browser again to `http://host_name:port/webapp/PaymentManager` where *host_name* is the host name of the machine running the Web Server for Payments, and *port* refers to the port number Payments is running on, and log on as Pat.
2. From the navigation frame, click **Approve**.
3. From the Approve page, select a box for one of the orders you placed.
4. Click **Approve Selected**. The Approve Results page displays the status of your approve request.
5. When your approval is complete, click **Return to the Approve screen**.

Two orders are still awaiting your approval. You could have approved them simultaneously from the Approve page for their full amount by clicking **Approve All**. However, to better demonstrate the approve functions, this tutorial describes how to work with each order individually.

Approving orders from the Order page

In this section, you will approve *part* of the total order amount of an order from the Order page. You may find it useful to approve only part of an order when some of the goods associated with the order are not available for delivery at order processing, for example, if merchandise is back-ordered.

1. From the Approve page, click the **Order number** for one of the remaining orders awaiting approval.
2. From the Order number page, you can view order details. Click **Approve** to approve this order.
3. The Order number approve page displays the following fields:

Table 15. Order number approve fields

Field name	Description
Currency	The type of currency used to place this order. This is a read-only field.
Order Amount	The total amount of the order expressed in the currency used to place the order. This is a read-only field.
Approved Amount	This read-only field displays zeros since no amount of the order has yet been approved.
Deposited Amount	This read-only field displays zeros since no amount has yet been approved or deposited.
Approval Amount	This is the total amount of the order.

Table 15. Order number approve fields (continued)

Field name	Description
Authorization Code	The authorization code returned from the manual offline authorization request process. The payment state is changed to approved.
Decline Reason	The decline reason returned from the manual offline authorization process. The payment state is changed to declined.
AVS Result Code	The Address Verification System result code.

Change the approval amount to 3.00. Optionally, specify an authorization reason to approve the amount or decline reason to indicate that approval was declined.

4. Click **Approve** to approve this order for three dollars. When the approval processing is complete, the Order page refreshes and displays approval status.

Approving orders with the sale function

Because you approved *part* of the last order you worked with, you still have two order entries in the Approve page. In this step, you will use the *sale* function to approve the remaining orders.

The sale function allows you to approve an order and move it directly into deposited state, bypassing approved state. The sale function automatically approves and deposits your order payment. Thus, you can think of sale as approve with auto-deposit. Use the sale function to expedite the delivery of goods to the buyer and guarantee your capture of funds, for example, when you are selling downloadable software or electronic information. However, with the sale function, you cannot set the authorization or decline reasons for the approval. Because sale merges approve and deposit into one transaction, it is also useful when you are charged on a per-transaction basis.

Do the following to approve an order with the sale function:

1. From the navigation frame click **Approve**.
2. From the Approve page, click **Sale All**. When processing is complete, the approval status displays for each order submitted for sale.
3. When your sale is complete, click **Return to the Approve Screen**.

Step 9: Depositing payments

The Deposit function allows you to deposit order payments. A single order number can have multiple payments associated with it. You may see the same order number appear multiple times in the same list, each time with different payment information.

To deposit a payment, do the following:

1. From the navigation frame click **Deposit**.
2. Select a box for one of the payments listed and click **Deposit Selected**. When the processing is complete, deposit status displays for the payment submitted for deposit.
3. When your deposit is complete, click **Return to the Deposit Screen**.

You can also deposit *part* of a payment. To deposit part of a payment, do the following:

1. From the Deposit page, click the **Payment number** for one of the payments awaiting deposit.
2. From the Payment page, click **Deposit**.
3. On the Order Payment page, change the deposit amount to 2.00 and click **Deposit**.

Step 10: Settling batches

A batch is a collection of payments and credits that are processed as a unit by a financial institution. A batch is associated with a merchant and an account. The payments that you deposited in the previous exercise will now appear in a batch. You must *settle* this batch to initiate processing by the financial institution. The financial institution is responsible for the transfer of funds once settle is complete.

To settle a batch, do the following:

1. From the navigation frame click **Batch Search**. Alternatively, you can click **Settle**.
2. In the Batch Search page you can enter the following information to narrow your search (note that for this tutorial, you do not need to complete these fields):

Table 16. Batch search fields

Field name	Description
Merchant	The name of the merchant whose batch you are searching for. If there are less than 500 merchants in the WebSphere Commerce Payments database, select the merchant name from the drop-down list. If there are more than 500 merchants in the WebSphere Commerce Payments database, type the name of a merchant.
Batch Number	The number that uniquely identifies the batch within the merchant. Assigned when the payment is deposited.
State	The state of the batch: <ul style="list-style-type: none"> • Open • Closed
Balance Status	The balance status of this batch: <ul style="list-style-type: none"> • Balanced The batch has been successfully balanced, that is, all totals agree. • Out of balance An unsuccessful attempt has been made to balance this batch, that is, all totals do <i>not</i> agree.
Payment Type	Identifies the payment type, or protocol, used to place the order, for example, OfflineCard.
Batch Open Date	Use the after and before fields to search for batches opened during the specified range in time: <ul style="list-style-type: none"> • After Specify a date to search for all batches opened on and after this date. • Before Specify a date to search for all batches opened on and before this date.
Batch Closed Date	Use the before and after fields to search for batches closed during the specified range in time: <ul style="list-style-type: none"> • After Specify a date to search for all batches closed on and after this date. • Before Specify a date to search for all batches closed on and before this date.

Table 16. Batch search fields (continued)

Field name	Description
Account	The account for which this order is being processed. If more than 500 accounts have been defined, type the account number in the entry field.

3. Click **Search**.

Tip: You can also use the before and after fields to narrow search results by excluding certain batches from the search. For example, you could search on all batches opened before 08/01/2003 and after 08/15/2003 thus excluding batches opened between 08/02/2003 and 08/14/2003

4. Click the batch number to view information about the batch.

5. Click **Batch Details** to see a detailed listing of all payments and credits in this batch.

6. Click **Settle** to settle the batch. When processing is complete settle status displays in the Settle Results page.

To prune outdated information, you can delete the settled batch by clicking **Delete** on the Settle Results page. When a batch is deleted, all ancillary information about that batch, that is, payments, credits, and cassette-specific data, is deleted, as well. If you need to retain all payment data, for example, for audit purposes, you should *not* delete a batch.

Step 11: Issuing a credit

Credits are issued against orders and can be given for any amount up to the total amount of the order.

To issue a credit, do the following:

1. To find the order for which you want to issue the credit, from the navigation frame click **Order Search**.
2. In the Order Search page, you can type the following (note that for this tutorial, you do *not* need to complete these fields):

Table 17. Order search fields

Field name	Description
Merchant	The name of the merchant whose order you are searching for. If there are less than 500 merchants in the WebSphere Commerce Payments database, select the merchant name from the drop-down list. If there are more than 500 merchants in the WebSphere Commerce Payments database, type the name of a merchant.
Order Number	A number assigned by the merchant that uniquely identifies the order.
State	The state of the order: <ul style="list-style-type: none"> • Requested • Ordered • Refundable • Canceled • Closed

Table 17. Order search fields (continued)

Field name	Description
Payment Type	Identifies the payment type, or protocol, used to place the order, for example, OfflineCard.
Order Date	Use the after and before fields below to search for orders opened during the specified range in time: <ul style="list-style-type: none"> • After Specify a date to search for all orders opened on and after this date. • Before Specify a date to search for all orders opened on and before this date.
Order Amount	<ul style="list-style-type: none"> • Currency The currency used to place this order. Select the currency type from the drop-down list. • Greater than Specify a value to retrieve all orders with order amounts that are greater than or equal to the value you specify. • Less than Specify a value to retrieve all orders with order amounts that are less than or equal to the value you specify.
Account	The account for which this order is being processed. If more than 500 accounts have been defined, type the account number in the entry field.

3. Click **Search**.
4. From the Order Search Results page, click an order number for an order in Refundable state to view the details of that order.
5. From the Order page, click **Credit** to create a credit against this order.
6. The Create Credit page displays the following information:

Table 18. Create credit fields

Field name	Description
Currency	The type of currency used to place this order. This is a read-only field.
Order Amount	The total amount of the order expressed in the currency used to place the order. This is a read-only field.
Approved Amount	The total amount of the order that has been approved expressed in the currency used to place the order. This is a read-only field.
Deposited Amount	The total amount of the order that has been deposited expressed in the currency used to place the order. This is a read-only field.
Credit Amount	This field must be completed by the Merchant Administrator with the total amount to be credited to the shopper.
Authorization Reason	The authorization code returned from the manual offline authorization request process. The credit state is changed to Refunded.
Decline Reason	The decline reason returned from the manual offline authorization process. The credit state changes to Declined.

Type the credit amount and click **Credit**.

When credit processing has completed, the Order page refreshes and displays the credit status. The newly created credit displays under **Credits**.

Step 12: Viewing batch totals

The last step of this tutorial is viewing daily batch totals. The WebSphere Commerce Payments reports function allows you to view *daily totals* for batches in a closed state.

To generate a daily batch totals report, do the following:

1. From the navigation frame click **Reports**.
2. From the Reports page, click **Daily Batch Totals**.
3. In the Batch Totals Report page, type the **Date** for which you would like a batch totals report. Leave this field blank to generate a report for the current date.
4. Type or select the **Merchant name**. If you do not type a merchant name, a list of all of the batches for the specified date displays. If there are more than 500 batches, only the first 500 batches display.
5. Click **Search**.

The Daily Batch Totals report computes the totals for all batches that were closed on the date specified in the **Search** page. Since you did not specify a search date, the report that was generated contains the current day's batch totals. These totals are computed on a per-currency basis, so there is one line per currency. Note that these totals include all payments and credits made for all payment types, not just those made through the OfflineCard Cassette.

You have just completed a day in the life of a Payments Administrator and a Merchant Administrator. While individual business models may vary, this tutorial outlines the basic path to establishing a working WebSphere Commerce Payments and demonstrates fundamental payment processing implemented through WebSphere Commerce Payments. For more information on specific fields in the WebSphere Commerce Payments user interface, see the online help.

Appendix D. National Language Support (NLS) information for WebSphere Commerce Payments

This chapter provides supplemental information on National Language Support (NLS) for WebSphere Commerce Payments. It is important that you review this information if you are installing a National Language Version (NLV) of WebSphere Commerce Payments.

NLS hints and tips

Adobe Acrobat Reader limitation

PDF files may not display correctly for some languages on the AIX platform. If you are unable to view or print WebSphere Commerce Payments PDF files (that is, the documentation files), you can view and print the files from a browser on the Windows[®] or Windows 2000 operating systems. This is a limitation of the AIX Adobe Acrobat Reader.

Code pages

If characters are not displayed correctly in Spanish, ensure that you are using code page ISO-8859-1.

WebSphere Commerce Payments user names

WebSphere Commerce Payments supports only those user names made up of characters in the following table. User names containing characters outside this set are not accessible.

This table lists the supported character set for assigning WebSphere Commerce Payments user names.

Table 19. Portable Character Set (PCS) for WebSphere Commerce Payments User Names and others

ASCII Hex value	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
Character	space	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
ASCII Hex value	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
Character	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
ASCII Hex value	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
Character	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
ASCII Hex value	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
Character	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
ASCII Hex value	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
Character	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
ASCII Hex value	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	
Character	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

Supported languages and locales

The following table lists each language and the corresponding locale that is supported by WebSphere Commerce Payments.

Table 20. Supported Languages and locales on WebSphere Commerce Payments

Language	Locale
Brazilian Portuguese	pt_BR
English	en_US
French	fr_FR
German	de_DE
Italian	it_IT
Japanese	Ja_JP
Korean	ko_KR
Simplified Chinese	zh_CN
Spanish	es_ES
Traditional Chinese	Zh_TW

Notices

Note to U.S. Government Users — — Documentation relating to restricted rights — — Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

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 IBM product, program, or service may be used. Any functionality equivalent to 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 the subject matter in this document. The furnishing of this document does not grant 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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chrome, Minato-ku
Tokyo 1061, Japan

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independent created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Lab Director
IBM Canada Ltd. Laboratory
8200 Warden Avenue
Markham, Ontario
L6G 1C7
Canada

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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 may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may 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.

This document may contain information about other companies' products, including references to such companies' Internet sites. IBM has no responsibility for the accuracy, completeness, or use of such information.

This product is based on the SET protocol.

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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. (C) Copyright IBM Corp. 1996, 2003. All rights reserved.

Trademarks

The IBM logo and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both:

- AIX
- CrossWorlds
- DB2 Extenders™
- DB2 Universal Database™
- IBM iSeries
- Intelligent Miner™
- Lotus®
- OS/400
- pSeries™
- S/390
- VisualAge®
- WebSphere
- xSeries®
- zSeries
- Redbooks
- SecureWay
- DFS
- VisualAge
- DB2
- AS/400

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

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

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

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

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



Printed in USA