Enhancements to IBM WebSphere Commerce Studio

WebSphere Commerce Studio and WebSphere Studio Application Developer Integration Guide

Version 5.4

Enhancements to IBM WebSphere Commerce Studio

WebSphere Commerce Studio and WebSphere Studio Application Developer Integration Guide

Version 5.4

Note:
Before using this information and the product it supports, be sure to read the information in the Notices section.

First Edition (March 2004)

This edition applies to the following products:

- IBM WebSphere Commerce Studio, Business Developer Edition for Windows NT and Windows 2000, Version 5.4 (Program 5724-A18)
- IBM WebSphere Commerce Studio, Professional Developer Edition for Windows NT and Windows 2000, Version 5.4 (Program 5724-A18)

and to all subsequent releases and modifications of the above listed products, 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. Publications are not stocked at the address given below.

IBM welcomes your comments. You can send comments about this publication by one of the following methods:

1. Electronically to following e-mail address:

   `torrcf@ca.ibm.com`
2. By mail to the following address:

   IBM Canada Ltd. Laboratory
   B3/KB7/8200/MKM
   8200 Warden Avenue
   Markham, Ontario, L6G 1C7
   Canada

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.

Before you begin

Document description

This *WebSphere® Commerce Studio and WebSphere Studio Application Developer Integration Guide* provides information regarding how to integrate the IBM® WebSphere Studio Application Developer Version 5.0 product into the WebSphere Commerce Studio Version 5.4 development environment. After performing this integration, you can use WebSphere Studio Application Developer Version 5.0 for your integrated development environment (IDE) rather than using VisualAge® for Java™.

Important: After switching your IDE to WebSphere Studio Application Developer, the development environment will contain WebSphere Commerce code that is at either the fix pack 5.4.0.6 level, or the fix pack 5.4.0.6 level with the Commerce Enhancement Pack (October 2002). You must ensure that your target WebSphere Commerce Server is also running at this level of code before deploying any customized code from the development environment to the target WebSphere Commerce Server.

Conventions and terminology used in this book

This book uses the following highlighting conventions:

Boldface type indicates commands or 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.

This icon marks a Tip — additional information that can help you complete a task.

Windows indicates information that is specific to WebSphere Commerce for Windows NT® and Windows® 2000.

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

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

400 indicates information specific to Commerce Suite, Pro Edition for the IBM @server™ iSeries™ 400® (formerly called AS/400®)

Professional indicates information specific to WebSphere Commerce Studio, Professional Developer Edition.

Business indicates information specific to WebSphere Commerce Studio, Business Developer Edition.

CEP indicates information that is specific to the Commerce Enhancement Pack.

DB2 indicates information specific to DB2 Universal Database™

Oracle indicates information specific to Oracle.

This book uses the following terms:

development machine
The machine on which you are going to develop customized code for WebSphere Commerce. This is the machine onto which WebSphere Commerce Studio is installed.

target WebSphere Commerce Server
The machine to which you will deploy your customized WebSphere Commerce code.

Path variables

This guide uses the following variables to represent directory paths:

- *VAJ_installdir*
  This is the installation directory for VisualAge for Java. An example path is
  ```
  C:\VAJava
  ```
- *WAS_installdir*
  This is the installation directory for WebSphere Application Server. An example path is
  ```
  C:\WebSphere\AppServer
  ```
- *WCStudio_installdir*
  This is the installation directory for WebSphere Commerce Studio. An example path is
  ```
  C:\WebSphere\CommerceServerDev
  ```

- *enhancements_installdir*
  This is the installation directory for the enhancements to WebSphere Commerce Studio. An example path is
  `C:\CommerceStudioEnhancments`
- *WSAD_installdir*
  This is the installation directory for WebSphere Studio Application Developer. An example path is
  `C:\Program Files\IBM\Application Developer`
- *workspace_dir*
  This is the directory for the WebSphere Studio Application Developer workspace. An example path is
  `C:\WCworkspace`

  Important: Due to a limitation related to the number of characters in a path (it must be less than 256 characters), you should not accept the default workspace directory when opening WebSphere Studio Application Developer for the first time. If the default directory is accepted, errors may be encountered later when attempting to test WebSphere Commerce code in the WebSphere test environment.

Knowledge requirements

This book should be read by Store Developers that want to use the WebSphere Studio Application Developer, Version 5.0 product to perform programmatic customizations for WebSphere Commerce applications. Store Developers that are performing programmatic extensions should have knowledge in the following areas:

- Java
- Enterprise JavaBeans component architecture
- JavaServer Pages technology
- HTML
- Database technology
- WebSphere Studio Application Developer, Version 5.0

Where to find more information

For more information related to WebSphere Commerce, refer to the following Web sites:

Professional `http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

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

Contents

Part 1. Installing and configuring the development environment

This part of the document provides information related to setting up a WebSphere Commerce development environment that uses WebSphere Studio Application Developer Version 5.0 as the integrated development environment (IDE).

Chapter 1. Introduction

This document provides information on the following topics:

- Installing software and configuring the development environment on a new development machine.
- Migrating a development machine that is currently configured to use the VisualAge for Java, Enterprise Edition component of WebSphere Commerce Studio, Version 5.4 to use WebSphere Studio Application Developer instead.
- Publishing stores within the WebSphere test environment component of WebSphere Studio Application Developer.
- Migrating customized WebSphere Commerce code that was created in VisualAge for Java to WebSphere Studio Application Developer.
- Deploying customized code from WebSphere Studio Application Developer to a target WebSphere Commerce Server.

This document refers to the *WebSphere Commerce Programmer's Guide* as a reference that provides details on the topic of code deployment. You should download the most recent copy of the *WebSphere Commerce Version 5.4 Programmer's Guide* from one of the following Web sites:

▶Professional `http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

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

On the preceding Web sites, click the link for user guides.

Note that once you have started to use WebSphere Studio Application Developer as your IDE, only certain parts of the current *WebSphere Commerce Programmer's Guide* remain applicable. In particular, the tutorials contained in the *WebSphere Commerce Programmer's Guide* no longer apply. They are written to introduce the WebSphere Commerce architecture and programming model within the context of VisualAge for Java, not within in the context of WebSphere Studio Application Developer.

In general, the first seven chapters of the *WebSphere Commerce Programmer's Guide* are still applicable, except for any VisualAge for Java specific step-by-step information. Also, most of the deployment information is still applicable, except for the steps about how to create JAR files. For more information about the new deployment process, refer to Chapter 10, "Deploying customized code," on page 57 in this document.

This book may contain references to the WebSphere Commerce online help. If you do not have WebSphere Commerce installed, you can download the online help from the preceding listed Web sites.

Development environment configurations

When the WebSphere Commerce Studio, Version 5.4 product (Professional Developer Edition or Business Developer Edition) was originally released, the IDE component of the suite of products was VisualAge for Java Enterprise Edition, Version 4.0. When WebSphere Commerce Studio was installed and the option to develop back-end business logic was selected, you had the option of creating a development database locally, or to create a development database at a later time on a remote machine. Additionally, the InFashion sample store could have been included in the development environment.

Now, with these enhancements to WebSphere Commerce Studio, you can switch the IDE from VisualAge for Java to WebSphere Studio Application Developer. This document describes how to setup a local development database.

Regarding the inclusion of a sample store, once WebSphere Studio Application Developer is installed and configured according to the instructions in this guide, you can then publish sample stores using Store Services running in the WebSphere test environment component of WebSphere Studio Application Developer.

Software and hardware prerequisites

This section describes the minimum software and hardware requirements that the development machine must meet. Additionally, the preferred configuration information provides a recommendation about the requirements for a more efficient development environment.

Minimum requirements

This section describes the minimum software and hardware requirements for your development machine.

- Pentium® III 700 MHz processor or higher
- 512 MB RAM minimum with at least 500 MB system swap
- Disk space requirements: 1 GB minimum
- Ensure that you have one of the following operating systems installed:
  - Windows 2000 Server Edition, or Advanced Server Edition with Service Pack 2 applied. You can obtain updates at the following Web site:

    `http://www.microsoft.com`
  - Windows NT Server Version 4.0 with Service Pack 6a installed on your development machine. You can obtain updates at the following Web site:

    http://www.microsoft.com

- Microsoft® Internet Explorer 5.5 or higher

Preferred configuration

This section describes software and hardware requirements to achieve a more efficient development environment.

- Pentium III 1 GHz processor or higher
- 1 GB RAM minimum with at least 1 GB system swap
- Disk space requirements: 5 GB minimum
- Ensure that you have one of the following operating systems installed:
  - Windows 2000 Server Edition, or Advanced Server Edition with Service Pack 2 applied. You can obtain updates at the following Web site:

    `http://www.microsoft.com`
  - Windows NT Server Version 4.0 with Service Pack 6a installed on your development machine. You can obtain updates at the following Web site:

    `http://www.microsoft.com`
- Microsoft Internet Explorer 5.5 or higher

Roadmap

This section provides guidance about which chapters of this document apply to various scenarios.

New development machines

If you are setting up a new development machine, or do not wish to preserve any work from a previous WebSphere Commerce Studio, Version 5.4 installation, begin reading this document in Chapter 1, “Introduction” and progress sequentially through the book. After going through all of the chapters in Part 1, “Installing and configuring the development environment,” skip to Part 3, “Code deployment,” on page 55.

Existing development machines

If you are migrating a development machine that already has WebSphere Commerce Studio, Version 5.4 installed, begin with reading Part 2, “Migration,” on page 31. This part refers to appropriate chapters in Part 1, “Installing and configuring the development environment.” Next, proceed to Part 3, “Code deployment,” on page 55.

Chapter 2. Installing prerequisite software

Before you can install the enhancements to WebSphere Commerce Studio, you must ensure that you have the following software installed:

- Database software:
  - DB2 DB2 Universal Database, Version 7.0 with fix pack 6. Ensure that you are using JDBC 2.

To determine if you are using JDBC 2, do the following:

1. Navigate to the SQLLIB\java12 directory.
2. If this directory contains an “inuse” file, open the file in a text editor to determine the level of JDBC that is currently in use.
3. If required, switch to JDBC 2, by doing the following:
   a. Ensure that no processes are currently using DB2® and that no DB2 process are running.
   b. Enter the following command:

      ```
      usejdbc2
      ```

  - Oracle Oracle 8.1.7.2.1
- WebSphere Studio Application Developer Version 5.0 general availability edition
  Refer to the following sections for information about installing WebSphere Studio Application Developer.

Installing WebSphere Studio Application Developer

WebSphere Studio Application Developer uses an installation wizard to guide you through the install process. When installing this product, make note of the following:

1. Insert the WebSphere Studio Application Developer, Version 5.0 general availability CD into your CD drive.
2. If the installation program does not automatically start, launch the `setup.exe` program located at the root of the CD.
3. Follow the onscreen instructions to install the product, but note the following points that apply to the installation procedure:
   - Ensure that you select to install the WebSphere Application Server Version 4.0 run-time environment. By default, this option is not automatically selected and you must explicitly select it.

- Do not install WebSphere Studio Application Developer while logged on as a user whose Windows NT or Windows 2000 user ID contains double-byte character set (DBCS) characters.
- By default, WebSphere Studio Application Developer is installed in the `system_drive\Program Files\IBM\WebSphere Studio\` directory (where *system_drive* is the drive where Windows NT or Windows 2000 is installed).
- If you override the default installation path, ensure that the installation path you select does not contain any DBCS characters.

4. You must apply the WebSphere Studio Application Developer Version 5.0.0.2 update and the WebSphere Studio Application Developer PTF 001 - General Fixes. For more information about how to apply these updates, refer to the following Web site:

   ```
   http://www.ibm.com/support/docview.wss?rs=457&context=SSBRLP
      &q=&uid=swg24004419&loc =en_US&cs=utf-8&lang=en
   ```

   Ensure that you read the install.html file for complete installation instructions.

Installing an Oracle database

If you will be using an Oracle database as your development database, you must create the database and the database user before you can install the enhancements.

For detailed steps about how to install this database, refer to the "Installing an Oracle database" chapter in the most recent version of the *WebSphere Commerce Studio Version 5.4 Installation Guide*. You can download this document from the following Web site:

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

Chapter 3. Installing the enhancements

Once the prerequisite software has been installed, you must install the enhancements to WebSphere Commerce Studio.

This installation process consists of the following high-level steps:

1. Installing the enhancements.
2. Creating a WebSphere Commerce development instance.

Each of these steps is described in more detail in subsequent sections.

Depending on your business requirements, you should have downloaded one of the following enhancement packages and saved it on your development machine:

- TLKT_BE_5406.zip
  Use this package if you are developing code for WebSphere Commerce Business Edition, at the fix pack 5.4.0.6 level
- TLKT_PRO_5406.zip
  Use this package if you are developing code for WebSphere Commerce Professional Edition, at the fix pack 5.4.0.6 level
- CEP TLKT_BE_5406_OCT02_CommerceEnhancementPack.zip
  Use this package if you are developing code for WebSphere Commerce Business Edition, at the fix pack 5.4.0.6 level with the Commerce Enhancement Pack (October 2002)
- CEP TLKT_PRO_5406_OCT02_CommerceEnhancementPack.zip
  Use this package if you are developing code for WebSphere Commerce Professional Edition, at the fix pack 5.4.0.6 level with the Commerce Enhancement Pack (October 2002)

For information about the download location for the preceding files, refer to one of the following Web sites:

Professional `http://www.ibm.com/software/webservers/commerce/wc_pe/support.html`

Business `http://www.ibm.com/software/webservers/commerce/wc_be/support.html`

Important: You can only install one version of the enhancements on your development machine. For information about determining which version of the enhancements is installed, refer to “Determining which version of the enhancements is installed” on page 72.

Installing the enhancements

After you have downloaded the enhancements package to your development machine, do the following:

1. At a command prompt, navigate to the directory into which you downloaded the enhancements package.
2. Unzip the package into a temporary directory. For example, unzip the package into C:\temp.
3. Navigate to this temporary directory and launch the setup.exe program. An installation wizard opens to guide you through the install.
4. Select the language for your installation and click **OK** to continue.
5. A Welcome window opens. Click **Next** to continue.
6. The License Agreement window opens. Review the terms of the license agreement and either accept or decline the agreement. If you accept the terms of the license agreement, the installation program will continue. If you decline the license agreement, the installation program will exit.
7. You are prompted to specify the directory to which the enhancements will be installed. Either accept the default location or into the **Directory Name** field, enter a new directory. Click **Next** to continue.
8. Summary information displaying the selected installation directory and the required disk space is displayed. Click **Next** to install the enhancements.
9. Progress of the installation process is displayed. When the installation is complete, click **Finish**.

Creating a WebSphere Commerce development instance

Note that you can have multiple workspaces to support various development projects. If you want to do this, you must create a separate instance to correspond to each workspace and use unique names for the both the instance and its database. If you are recreating an instance using an existing instance name, ensure that you perform the instance cleanup steps before recreating the instance (refer to “Removing an old instance” on page 13 for more information).

Each development instance requires approximately 400MB of free space on your hard drive.

When you create your development instance, according to the instructions contained in this section the following tasks are accomplished:

1. Workspace preferences are set.
2. The WebSphere Commerce development instance is created.
3. The local WebSphere Commerce development database is created.

4. The WebSphere Commerce workspace is imported into WebSphere Studio Application Developer.

To create your development instance, do the following:

1. Start WebSphere Studio Application Developer.

   Important: You must change the default workspace directory. When WebSphere Studio Application Developer is started for the first time, you are prompted to select a directory for your workspace. By default, the default workspace directory is

   ```
   C:\Documents and Settings\win_user\My Documents\IBM\wsad\
   workspace
   ```

   where `win_user` is the Windows user ID with which you are currently logged on. You *must not* accept these settings. Change the directory path to a path that is much shorter, for example, set it to the following:

   ```
   C:\WCworkspace
   ```

   The preceding directory is referred to as the `workspace_dir` directory in this document.

   Note: If you are prompted to select your workspace, select *workspace_dir*. If you are not prompted, you can start the application and use the correct workspace, by doing the following:

   a. At a command prompt, navigate to the *WSAD_installdir* directory.
   b. Enter the following command:

      ```
      wsappdev -data workspace_dir -showlocation
      ```

   c. As another option, you can enter the following command so that you will be prompted for your workspace location:

      ```
      wsappdev -setworkspace
      ```

2. Set your workspace preferences, by doing the following:
   a. From the **Window** menu, select **Preferences**.
      The Preferences window opens.
   b. Click **Import**. When prompted, select the preferences file that is found as follows:

      ```
      enhancements_installdir\toolkit\workspace\wsad.epf
      ```

      Click **OK**.
   c. From the **Window** menu, select **Preferences** again.
   d. In the left pane, select **Workbench**, then in the right pane, clear the **Perform build automatically on resource modification** check box.
   e. In the left pane, select **Validation**, then in the right pane, clear the following check boxes

- **HTML Syntax Validator**
- **JSP Compilation Validator**
- **WAR Validator**

and click **OK**.

3. Close WebSphere Studio Application Developer.
4. At a command prompt, navigate to the following directory:

   ```
   enhancements_installdir\bin
   ```
5. Enter the following command:

   ```
   createInstance workspace_dir instanceName dbName dbUser dbPassword dbType
   ```

   where
   - *workspace_dir* is the directory for WebSphere Commerce workspace.
   - *instanceName* is the name of your WebSphere Commerce development instance.
   - *dbName* is the name of your development database.
   - *dbUser* is the database user.
   - *dbPassword* is the password of your database user.
   - *dbType* is the type of database you are using for your development database. Valid options are `DB2` and `Oracle`.

   The following is an example of this command with sample values provided:

   ```
   createInstance C:\WCworkspace wcdev wcdev db2admin db2admin DB2
   ```

   The createInstance script creates your instance and imports and configures your workspace. Expect this step to take at least 30 minutes to complete. You can follow its progress in the *enhancements_installdir*\logs\createInstance.log file. The process has finished when "Instance creation has completed." is displayed in the log file.

If you receive an error indicating that the specified workspace is incorrect, do the following:

1. At a command prompt, navigate to the *WSAD_installdir* directory.
2. Enter the following command:

   ```
   wsappdev -setworkspace
   ```
3. Open WebSphere Studio Application Developer and select *workspace_dir* as the location of the workspace.
4. Close WebSphere Studio Application Developer and rerun the createInstance command.

The next step is to start WebSphere Studio Application Developer and publish a store using Store Services. Refer to Chapter 4, "Publishing stores using Store Services," on page 15 for more information.

Removing an old instance

To remove an old instance, do the following:

1. Drop the development database that was specified when you ran the createInstance command.
2. Delete the instance directory. This directory is called:

   enhancements_installdir`\instances\`*instanceName*

   where *instanceName* is the name of the instance you are removing.
3. Restore the *enhancements_installdir*\instances\wcs_instances.bak file to *enhancements_installdir*instances\wcs_instances. The result will be a file that has no entries after the [instance] tag.
4. Delete your workspace directory.

Chapter 4. Publishing stores using Store Services

This chapter provides information specific to publishing WebSphere Commerce stores, using Store Services running within the WebSphere test environment component of WebSphere Studio Application Developer.

Publishing a store

CEP **Important:** The recommended sample store for viewing business-to-consumer (B2C) functions is the new FashionFlow sample store. This sample store was introduced in the Commerce Enhancement Pack (October 2002). FashionFlow merges the features of all of the WebSphere Commerce B2C sample stores (WebFashion, NewFashion, InFashion, and WebAuction). It also incorporates some new features. For more information about the FashionFlow sample store, refer to the *FashionFlow Sample Store Guide*.

Store Services running within the WebSphere test environment is used to publish WebSphere Commerce stores that are used in development.

Publishing tips
When publishing stores using Store Services running in the WebSphere test environment, keep the following points in mind:

1. Business If you want to publish both the FashionFlow store and the NewFashion store, they must be published under different store owners. For example, publish FashionFlow under the Seller organization and NewFashion under the Default organization.
2. Professional FashionFlow and NewFashion cannot coexist. Use FashionFlow, as it contains all of the functionality of NewFashion, as well as additional functionality.

The following steps describe how to publish a WebSphere Commerce sample store in the WebSphere test environment:

1. Open WebSphere Studio Application Developer (**Start > Programs > IBM WebSphere Studio > Application Developer 5.0**).

   Note: If you are prompted to select your workspace, select *workspace_dir*. If you are not prompted, refer to "Changing workspaces" on page 69 for more information.
2. Open the Server perspective (**Window > Open Perspective >(Other >) Server**)

3. Right-click the **WebSphereCommerceServer** server and select **Start**.
4. Open a Web browser, and then open Store Services by entering the following URL:

   ```
   http://localhost/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=devtools.Logon
   ```
5. ▶ Business (Optional) Create an organization or organizational unit for the site, and then create users, as described in the WebSphere Commerce online help. Note, if you need to launch the Administration Console, use the following URL:

   ```
   http://localhost/webapp/wcs/tools/servlet/ToolsLogon?
      XMLFile=adminconsole.AdminConsoleLogon
   ```
6. Logon to Store Services. You can logon using the wcsadmin user ID (the initial password is wcsadmin, however it should be changed the first time it is used), the user ID of another site administrator, or the user ID of a store administrator that was created in step 5.
7. Create a new store archive, based upon one of the sample store archives, as described in the WebSphere Commerce online help.
8. From the **Store Archive** list in the Store Services pane, select the store archive you created in step 7 and click **Publish**.
9. Select all of the publishing options. Ensure that the path values are specified as follows:
   - Web application document root

     workspace_dir`\wcstores\Web content`
   - Application properties path

     workspace_dir`\wcstores\Web content\WEB-INF\classes`

   where *workspace_dir* is the location of your workspace directory.
10. When ready to publish, click **OK**.
11. While the store is being published, the browser returns to the Store Archive list page. On this page, the publishing state is reflected in the Publish status column. Click **Refresh** to update the status.

    Note: The logging for the Loader package has been configured so that trace information is not written, by default. This logging can be configured in the `WCALoggerConfig.xml` file, but turning the trace on will negatively impact performance of the publishing process.
12. When publishing has completed, select your store archive from the list and click **Publish Summary** to view the results of the publishing process. If an error occurred during the publish, a window displays more information about the error. For more information, refer to the "Troubleshooting publishing" topic in the WebSphere Commerce online help.
13. After a successful store publish, use the Navigator View in WebSphere Studio Application Developer to browse the **wcstores** Web project.

Expand the **wcstores** Web project, then right-click the **Web Content** subfolder and select **Refresh**. The JSP templates, HTML files and other Web assets should now be available in the following folder:

```
wcstores\Web Content\storeDir
```

where *storeDir* is the name of your new store.
The properties files for the store should now be available in the following directory:

```
wcstores\Web Content\WEB-INF\classes\storeDir
```

If rebuilding the `wcstores` Web module, the `classes` directory is emptied and created from the `source` folder. As a result, you should copy *storeDir* to the `source` folder.

---

14. Launch your store using one of the following methods:
    a. Using the **Launch store** button in Store Services.
    b. By entering the following URL in a Web browser:

       ```
       http://localhost/webapp/wcs/stores/servlet/storeDir/index.jsp
       ```

       where *storeDir* is the directory name for your store.

Chapter 5. Understanding payment options

When running stores within the WebSphere test environment component of WebSphere Studio Application Developer, you have different options for processing payments. If you are using a business-to-consumer store, you can either use a test payment method, or call out to a remote Payment Manager. The test payment method is only to be used for testing purposes, and was only designed to allow you to complete a purchase. By default, the test payment method is configured when you create your WebSphere Commerce development instance.

The test payment method cannot confirm that payment was accepted, cancel payments, deposit funds, or perform refunds. If you wish to perform these actions you must install WebSphere Payment Manager on a remote machine.

If you are using a business-to-business store, you can either use the Credit Line payment option, or call out to a remote payment manager.

For business-to-consumer stores, this chapter provides information about how to switch from using the test payment method to using a remote Payment Manager.

For business-to-business stores, this chapter provides information about how to enable the Credit Line payment option for use in the development environment.

Using Credit Line in a business-to-business store

This section provides instructions to use the Credit Line payment option in your business-to-business store for use in your development environment. Note that these instructions are provided so that you can easily complete a purchase within the development environment. For information on using this payment option outside of the development environment, ensure that you refer to the WebSphere Commerce online help.

To use the Credit Line payment method within your development environment, do the following:

1. Publish a store based upon the ToolTech sample store.
2. Create a new registered user. Be sure to specify the following options when creating this user:
   a. In the **Buyer organization** field, enter `Buyer organization b`.

b. Fill in the other mandatory fields with appropriate values for your testing scenario.
c. Click **Submit**.

3. Launch the Adminstration Console and logon as a site administrator. (Refer to “Links to WebSphere Commerce tools” on page 69 for more information about accessing this tool.)
4. From the **Approvals** menu, select **Approve requests**.
5. Select the user you created in step 2. Click **Approve**, then click **OK**.
6. Logon to your business-to-business store as the user created in step 2. Browse the catalog and select a product. When selecting the product, select one that uses the “4567” contract.
7. When completing the purchase, select **Credit Line** as the payment method.

Using a remote Payment Manager with the WebSphere test environment

If the test payment method is insufficient for your development needs, you can configure your development environment to work with a remote Payment Manager. In doing so, orders placed in your stores running in the WebSphere test environment component of WebSphere Studio Application Developer can be processed.

Note that once you configure your development environment to use a remote Payment Manager, all stores that you have will use the remote Payment Manager. If you have previously configured your development environment to use the test payment method, the test payment method will be ignored once you have configured the development environment to use the remote Payment Manager.

Configuring your development environment to use a remote Payment Manager involves the following steps:

1. Completing your WebSphere Studio Application Developer configuration steps. You must be able to publish (as described in Chapter 4, “Publishing stores using Store Services,” on page 15) and launch your store within the WebSphere test environment before continuing to the next step.
2. Configuring Payment Manager to use the WCSRealm.
3. Restarting the WebSphere test environment.

Configuring your development environment to use a remote Payment Manager

This section describes the step-by-step details for configuring your development environment to use a remote Payment Manager.

Preparation

It is assumed that you have successfully installed and configured Payment Manager on a machine separate from your development machine. For complete details on installing and configuring Payment Manager, refer to the *IBM WebSphere Payment Manager for Multiplatforms Installation Guide*.

Configuring your WebSphere Commerce Studio instance to use a remote Payment Manager

To configure your instance to use a remote Payment Manager, do the following on your development machine:

1. Open the *enhancements_installdir*`\instances\`*`instance_name`*`\xml\`*`instance_name`*`.xml` file in a text editor, and do the following:
   a. Search for the `PaymentManager` element within the file, and edit the following attributes:
      - UseNonSSLPMClient="1"
      - Hostname="*host_name*"
      - WebServerPort="*port_number*"
      - UseExternalPM="false"
      - ProfilePath="*profile_path*"
      - If you are using a SOCKS server, do the following:

        Note: If you do not know if you are using a SOCKS server, please consult your network administrator.
        - Set UseSocksServer="true"
        - Add SocksHostname="*socks_server_hostname*"
        - Add SocksPort="*socks_port*"

      where
      - *host_name* is the fully qualified host name of the Payment Manager machine.
      - *profile_path* is the full path name of the directory where the standard WebSphere Commerce Payment Manager Cashier Profiles are located. By default, this should be:

        `enhancements_installdir``\instances\`*`instance_name`*`\`
        `xml\payment`
      - *port_number* is the Web server port that Payment Manager uses for non-SSL transactions. The default port number is 80.
      - *socks_server_hostname* is the fully qualified host name of your SOCKS server.
      - *socks_port* is your SOCKS port number.

Restart the servlet instance.

Configuring Payment Manager to use the WCSRealm

To use the WCSRealm on a remote machine where Payment Manager is installed, you must manually configure Payment Manager through the Administration Console. To manually configure your system to use WCSRealm, do the following:

1. Open the WebSphere Application Server Administration Console on the Payment Manager machine.
2. Navigate to and access the WebSphere Payment Manager application server by doing the following:
   a. Expand the WebSphere Administrative Domain.
   b. Expand Nodes.
   c. Expand the *host_name*.
   d. Expand **Application Servers**.
   e. Select WebSphere Payment Manager. Click the **Stop** button to stop the WebSphere Payment Manager application server.
3. After the WebSphere Commerce Payments application server has stopped, select the **JVM Settings** tab page of the application server. Scroll down to the **System Properties** box, select the `wpm.RealmClass` system property, and do the following:
   - Replace the `com.ibm.etill.framework.payserverapi.PSDefaultRealm` value with `com.ibm.commerce.payment.realm.WCSRealm`

   Click **Apply**.
4. If Payment Manager is enabled for SSL, you must also use the WebSphere Application Server administrative console to add *:443 entries to the default host window under Virtual Hosts.

   Under the WebSphere Administrative Domain, click Virtual Hosts and on the General tab for the default host, do the following:
   a. Add *:443 as a Host Alias.
5. In the Payment Manager install directory, open the `WCSRealm.properties` file to change the value of the `WCSHostName` property to the fully-qualified host name of the WebSphere Commerce Studio machine with the WebSphere test environment instance. In addition, while editing the `WCSRealm.properties` file, add the following lines to the bottom of the file:

   ```
   WCSWebServerPort=webserverPortNumber
   UseNonSSLWCSClient=1
   ```

   where *webserverPortNumber* is the HTTP transport that is used by the WebSphere Commerce development instance running in WebSphere Studio Application Developer. To verify this value, do the following:
   a. Open WebSphere Studio Application Developer and switch to the Server perspective.

   b. Double-click the **WebSphereCommerceServer** server.
   c. Click the Ports tab and verify the HTTP transport number..

   Save the `WCSRealm.properties` file.

6. Start the Payment Manager Application Server, within the WebSphere Application Server Administration Console.

   If you added *:443 entries to the default host window, you also need to stop and restart WebSphere Application Server.

7. Start Payment Manager:
   a. Open a command prompt.
   b. Change to the Payment Manager install directory.
   c. Run the following command: `IBMPayServer`.

   You will be prompted to enter your Payment Manager password, which is the password of the user you specified when connecting to the Payment Manager database.

   Note: Once these updates to your Payment Manager machine are complete, you may only use Payment Manager when WebSphere Commerce is running under the WebSphere test environment. Payment Manager relies on WebSphere Commerce to authenticate the Payment Manager administrator.

Creating a merchant, brand, and account in Payment Manager

Before using the remote Payment Manager with a WebSphere Commerce store running in the WebSphere test environment, you must create a merchant, brand, and account on the Payment Manager server. To configure your Payment Manager in this manner, do the following:

1. Ensure that WebSphere Commerce is running under the WebSphere test environment.
2. Log in to Payment Manager using the `wcsadmin` user ID and the password. You can access the Payment Manager log in page by entering the following URL in a browser:

   http://*host_name*/webapp/PaymentManager
3. From the left hand panel, click **Merchant Settings**. In the right hand window, click **Add a Merchant**.
4. Enter a Merchant Name for your sample store. The Merchant Number must be the same as the storeId for your store (e.g. 10001). Check the **OfflineCard** check box and click **Create Merchant**. A "Merchant created successfully" message should appear.
5. In the left hand panel, click Merchant Settings again. In the right hand window, beside the name of your new merchant, the green cassette status icon (under the OfflineCard column) indicates that the OfflineCard cassette

is enabled and running for that merchant. Click on the green icon. You must now create an account for each currency supported by your store, by clicking on **Accounts**.

6. Click **Add an Account** Enter an Account Name and an Account Number. The account number must be unique within that store. Select the appropriate currency from the drop down list. Click **Create Account**.
7. In the Accounts window, click on the name of your new account. Then click on **Brands**
8. Click on **Add a Brand** Enter a Brand Name (e.g. VISA) and click **Create Brand**.
9. Repeat for all other Brands to be used with this account. Repeat the above three steps to add accounts for all other currencies to be supported by your store.

Removing orders that used the TestPaymentMethod

To use a remote Payment Manager, you must remove any orders that were placed using the TestPaymentMethod. To remove these orders, do the following:

- Open a DB2 or Oracle session and connect to your instance database. Enter the following statements (Oracle users must include a semicolon at the end of each command):

  ```
  delete from orders where orders_id in ( select orders_id
  from ordpaymthd where policy_id in (select policy_id from
  policy where policyname='TestPaymentMethod')) and storeent_id = your_storeent_id

  delete from policy where policyname='TestPaymentMethod'
  ```

  where *your_storeent_id* is the unique identifier for the store in which the orders were placed.
  Oracle users must enter the following command after running these SQL statements:

  ```
  commit;
  ```

Verifying your remote Payment Manager

To verify that your store can successfully use the remote Payment Manager, you must launch your store and attempt to complete a purchase in that store. As such, you should do the following:

1. Ensure that your server instance for WebSphere Commerce is running.
2. Launch your store in a browser. For example, you might enter a Web address similar to the following:

   ```
   http://localhost/webapp/wcs/stores/servlet/storeDir/index.jsp
   ```

   where *storeDir* is the name of your store.
3. To place an order using Payment Manager, do the following when providing credit card information:

a. From the **Credit card type** drop-down list, select the brand name that you have entered on step number 8 on page 24. For example, **VISA**.
b. In the **Card number** field, enter a 16 digit number. For example, `4111111111111111` (4 followed by 15 ones).
c. Set the expiration month and year to a future date.

Chapter 6. Developing code for iSeries

If your target deployment WebSphere Commerce Server is running on iSeries, specific steps need to be taken in order to be able to deploy your customized code to that platform. This chapter provides information about these steps.

Development environment for iSeries

In short, a special development environment is not required in order to create customized code for iSeries. The development workstation is setup following the same procedure that is outlined in Chapter 2, “Installing prerequisite software,” on page 7 and in Chapter 3, “Installing the enhancements,” on page 9. The local development database used should be DB2. Using this configuration, customized code can be created and tested using the local DB2 database and local WebSphereCommerceServer test server.

After testing determines that the customized code functions to your satisfaction within the context of the test server, you must then deploy it to a target WebSphere Commerce Server running on the iSeries platform. In order to account for differences between the database on the Windows and iSeries platforms, a WebSphere Commerce enterprise bean conversion tool is provided. Steps on how to use this tool are provided in the next section.

Overview of the WebSphere Commerce enterprise bean conversion tool

At a high-level, using the conversion tool involves the following steps:

1. On the development machine, create code in the test and development workspace until the code functions as desired.
2. Using the file system, create a replica of the test and development workspace.
3. Run the WebSphere Commerce enterprise bean conversion tool.
4. Import the output from the conversion tool into the replica workspace.
5. Open WebSphere Studio Application Developer using the replica workspace and generate deployed code for any EJB groups that you have modified.
6. Deploy the customized code to your target WebSphere Commerce Server running on iSeries.

These steps are shown in the following diagram:

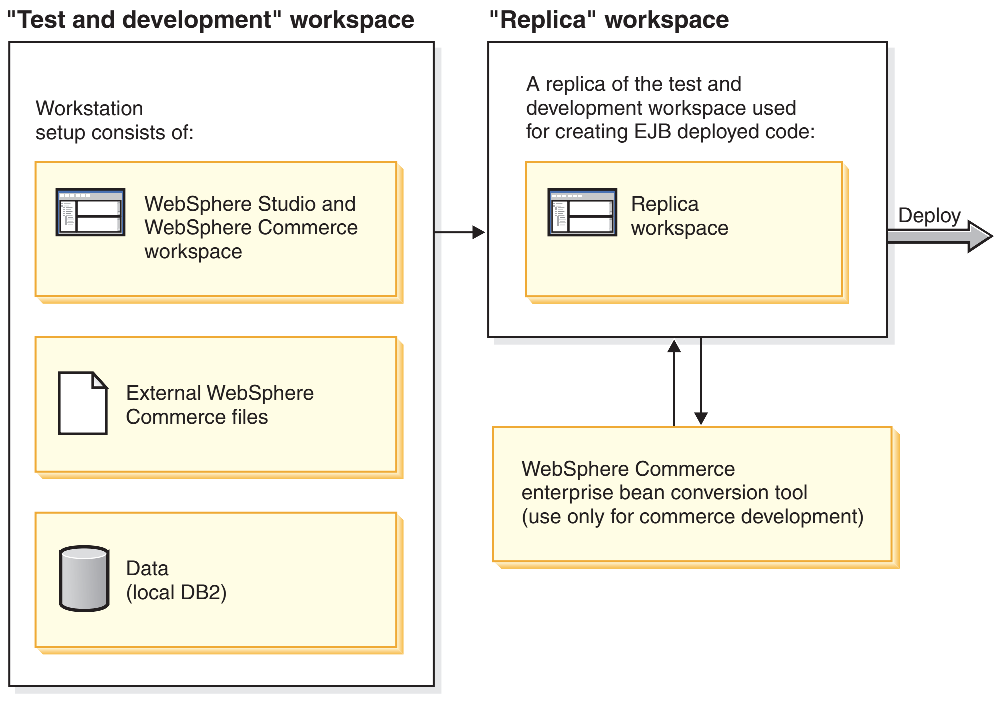
"Test and development" workspace
Workstation setup consists of:
WebSphere Studio and WebSphere Commerce workspace
External WebSphere Commerce files
Data (local DB2)
"Replica" workspace
A replica of the test and development workspace used for creating EJB deployed code:
Replica workspace
Deploy
WebSphere Commerce enterprise bean conversion tool (use only for commerce development)

Figure 1.

Using the WebSphere Commerce enterprise bean conversion tool

To use this conversion tool, do the following:

1. Create a copy of your test and development workspace directory. For example, make a copy of the entire C:\WCworkspace directory, and call it C:\tempWorkspace.
2. Using a command prompt, navigate to your test and development workspace directory (for example, navigate to C:\WCworkspace).
3. Temporarily set the class path to include the wcsconvert.jar JAR file, by entering the following command:

   ```
   set CLASSPATH=%CLASSPATH%;enhancements_installdir\toolkit\
      tools\wcsconvert.jar;
   ```

   Note, the line break in the preceding class path is for presentation purposes only.
4. Run the meta data conversion program, by entering the following command:

```
WSAD_installdir\eclipse\jre\bin\java
com.ibm.commerce.metadata.conversion.ConvertMetaData  "workspace_dir"
"iSeriesPropMapFile" "outputDir"
```

where

- *workspace_dir* is the directory of your test and development workspace
- *iSeriesPropMapFile* is the iSeries property map file
- *outputDir* is the directory where the converted meta data is placed.

The following is an example of the command with values filled in (line breaks are for presentation purposes only):

```
C:\Program Files\IBM\WebSphere Studio\eclipse\jre\bin\java
com.ibm.commerce.metadata.conversion.ConvertMetaData "C:\WCworkspace"
"C:\CommerceStudioEnhancements\toolkit\tools\as400.mapping"
"C:\mappingOut"
```

5. Once the conversion tool has finished, navigate to the directory you specified where the converted meta data should be placed. For example, navigate to C:\mappingOut.
6. From this output directory, copy the files corresponding the EJB groups that you modified into the replica workspace (for example, C:\tempWorkspace). Overwrite the existing meta data.
7. Open WebSphere Studio Application Developer using the replica workspace.

   Note: If you are prompted to select your workspace, select the directory for your replica workspace (for example, C:\tempWorkspace). If you are not prompted, refer to “Changing workspaces” on page 69 for more information.
8. For each EJB group that contains modified code, you must generate deployed code, as follows:
   a. From the **Window** menu, select **Show View > Navigator**.
   b. Navigate to the first EJB group that contains modified code. Right-click on this group and select **Generate Deployed and RMIC code**.
   c. In the Generate Deployed and RMIC Code window, select all of the enterprise beans and click **Finish** to generate the code.
   d. Repeat these steps for each EJB group that contains modified code.

Once the deployed code is generated, you can follow the regular instructions (refer to Chapter 10, “Deploying customized code,” on page 57) to deploy the customized code to your target WebSphere Commerce Server. Additionally, after the code has been deployed, you can remove the replica workspace from your workstation, if desired.

Part 2. Migration

There are two distinct areas that are affected when you are migrating from a development environment that uses VisualAge for Java, to one that uses WebSphere Studio Application Developer. These are:

1. Migrating your original software stack, WebSphere Commerce development instance and development database from the version 5.4 level to the version 5.4.0.6 (or version 5.4.0.6 with Commerce Enhancment Pack (October 2002)) level.
2. Transitioning customized code from VisualAge for Java to WebSphere Studio Application Developer.

Chapter 7. Inplace migration

The inplace migration upgrades your existing WebSphere Commerce database schema and WebSphere Commerce development instance information to the version 5.4.0.6 (or version 5.4.0.6 with Commerce Enhancment Pack (October 2002)) level. It also creates a WebSphere Commerce workspace in WebSphere Studio Application Developer.

Prerequisites for migration

Before starting the migration steps, you must ensure that your development machine meets the following requirements:

1. The prerequisites described in Chapter 2, "Installing prerequisite software," on page 7.
2. WebSphere Commerce Studio Version 5.4 (either the Professional or Business Developer Edition) must be installed. When this product was installed, you must have selected the following installation options:
   - Develop store back-end logic using VisualAge for Java
   - Create database

Migration steps

Once you have satisfied the prerequisites, you are ready to proceed with the migration steps, as follows:

1. Install the WebSphere Commerce fix pack 5.4.0.6. Refer to one of the following Web sites for information about this fix pack:
   - ▶Business `http://www.ibmc.com/software/webservers/commerce/wc_be/support.html`
   - ▶Professional `http://www.ibmc.com/software/webservers/commerce/wc_pe/support.html`
2. Install the enhancements, as described in Chapter 3, "Installing the enhancements," on page 9.
   Important: When running the createInstance command, ensure that you specify the name of your existing WebSphere Commerce development instance, as well as for your existing development database (note, this database must be local).
3. Publish a store in WebSphere Studio Application Developer, as described in Chapter 4, "Publishing stores using Store Services," on page 15. Launch this new store and verify that your environment works well.

4. Transition customized code from VisualAge for Java to WebSphere Studio Application Developer. Details about how to transition your code are provided in Chapter 8, “Code transition,” on page 35.
5. Test your code on the WebSphereCommerceServer test server running in WebSphere Studio Application Developer.

Chapter 8. Code transition

This chapter provides the detailed steps about how to migrate customized WebSphere Commerce code that was created using VisualAge for Java to the new WebSphere Studio Application Developer environment.

Before you begin the step in this chapter, you must have completed the steps in Chapter 7, "Inplace migration," on page 33.

Differences between VisualAge for Java and WebSphere Studio Application Developer

Before you transition code from your VisualAge for Java workspace to the WebSphere Studio Application Developer, it is important to understand the differences between the two software applications. The following list highlights the main differences between VisualAge for Java and WebSphere Studio Application Developer:

- The supported Enterprise JavaBeans (EJB) specification level used by WebSphere Commerce has changed from 1.0 to 1.1.
- For Web applications, the JSP level remains at 1.2.
- For Web applications, the Servlet level remains at 2.3.
- The level of the Java 2 platform that is supported has changed from 1.2 to 1.3 (The compiler can target 1.4 code generation, but the WebSphere Application Server run-time environment is still 1.3.)
- VisualAge for Java version control and the propriety source code repository have been replaced by support for software configuration management (SCM) plug-ins. The VisualAge for Java Tools API has been replaced by the WebSphere Studio Workbench plug-in architecture.
- The VisualAge for Java XML tools have been replaced by WebSphere Studio Application Developer XML tools.
- The VisualAge for Java project concept has been replaced by multiple types of WebSphere Studio Application Developer projects.
- The convertors that were availalbe in the VisualAge for Java access beans for enterprise beans (not data access beans) are not available in WebSphere Studio Application Developer access beans for enterprise beans. In WebSphere Studio Application Developer, use enterprise bean convertors and composers in the underlying enterprise bean instead. For more information about enterprise bean convertors and composers, refer to the WebSphere Studio Application Developer online help.

Transitioning extended or modified WebSphere Commerce business logic and code

As you transition customized code from VisualAge for Java to WebSphere Studio Application Developer, a number of steps are required. These steps are as follows:

- Exporting your Java files and project resource files from VisualAge for Java.
- Starting WebSphere Studio Application Developer and creating new projects to contain your code.
- Importing your Java and project resource files into WebSphere Studio Application Developer.
- Migrating new enterprise beans that you created in VisualAge for Java into WebSphere Studio Application Developer.
- Migrating any public WebSphere Commerce entity beans that you have modified from VisualAge for Java into WebSphere Studio Application Developer.
- Setting up your test environment and testing your migrated application.

Exporting your Java files and project resource files from VisualAge for Java

The first step in transitioning code to WebSphere Studio Application Developer involves exporting Java files and project resource files from VisualAge for Java.

Notes:

1. There is no support for the bulk migration of versioned projects and resources from the VisualAge for Java repository. You can, however, migrate projects and resources that are in your VisualAge for Java workspace. If you want to migrate a versioned copy of a project or resource into WebSphere Studio Application Developer, you must bring it into your VisualAge for Java workspace and then migrate it.
2. If your project contains more than one kind of data (for example, enterprise beans and Java source code files), you should split up your data into different JARs based on their type.

To export your projects to a JAR file, follow these steps:

1. If the projects that you want to export are not currently in your VisualAge for Java workspace, add them to the workspace.
2. In the VisualAge for Java Workbench window, select you project or projects, right-click, and click **Export**.
3. Select the **Jar file** radio button and click **Next**.
4. Type the name of the JAR file.
5. Select the **.java** check box to export your Java files and the **resources** check box to export your resource files.

6. Fill in the other fields as required. Refer to the VisualAge for Java online help for more information on how to complete the fields.

Starting WebSphere Studio Application Developer and creating new projects to contain your code

After you have exported your Java files and project resource files from VisualAge for Java, the next step in transitioning your code WebSphere Studio Application Developer is to start WebSphere Studio Application Developer and create appropriate projects. The following is a set of general migration guidelines to help you decide the kind of WebSphere Studio Application Developer project into which you should import your files. It is strongly recommended that you read the WebSphere Studio Application Developer online help and become familiar with the different kinds of WebSphere Studio Application Developer projects before you create any projects or import any code.

- If your code is part of a Web application, you should import the code into a Web project as follows:
  - Import all Java files, such as a controller commands, task commands, and data beans, into the following directory in the WebSphere Commerce workspace:

    ```
    wcstores\Java Source
    ```

    The proper hierarchy based on package statements will automatically be created by WebSphere Studio Application Developer.

    Note: If instead of importing source code you are importing binary files (.class files), you should import the files into the following directory:

    ```
    wcstores\WEB-INF\classes
    ```
  - Import all resource files, such as JSP pages, into the following directory:

    ```
    wcstores\Web content
    ```

    Note: If your assets are specific to a particular store, place them in a store-specific directory within the preceding directory, as appropriate to your application.
- If your code is enterprise beans, you should import the code into an enterprise bean project. If the beans are new enterprise beans that you have created, you should create a new EJB 1.1 project (referred to as *CustomizedDataEJBProject*) and import all of your beans into that project. If you have modified existing WebSphere Commerce entity beans, you should not create a new project, as the customized code will be merged into existing projects.

Importing your Java and project resource files into WebSphere Studio Application Developer

After starting WebSphere Studio Application Developer and creating new projects to contain your code, the next step in transitioning your code to WebSphere Studio Application Developer is to import your Java and project resource files into WebSphere Studio Application Developer.

Note: When you import your files into WebSphere Studio Application Developer, ensure that they are imported to the appropriate directory. It is recommended that you read the WebSphere Studio Application Developer online help and become familiar with the different kinds of WebSphere Studio Application Developer projects before importing your code. This will help you determine which folders should contain which kind of code.

To import your Java and project resource files into WebSphere Studio Application Developer, follow these steps:

- Open WebSphere Studio Application Developer and switch to the Resource perspective.
- Select **File** > **Import** > **Zip file**. Click **Next**.
- Browse to the appropriate JAR file.
- Select the files you want to import and the project or folder you want to contain your files.
- After the import is complete, rebuild the project to compile your code.

Migrating your enterprise beans into WebSphere Studio Application Developer

After importing your Java and project resource files into WebSphere Studio Application Developer, the next step in transitioning your code to WebSphere Studio Application Developer is to migrate your enterprise beans into WebSphere Studio Application Developer. This task involves three subtasks: exporting the enterprise beans from VisualAge for Java, importing the enterprise beans into WebSphere Studio Application Developer, and generating deploy code.

Exporting your enterprise beans from VisualAge for Java

1. In VisualAge for Java 4.0, add the **Export Tool for Enterprise Java Beans 1.1** feature to the workspace if it is not already present.
2. On the EJB page of the Workbench, right-click on your enterprise bean group or groups and click **Export** > **EJB 1.1 JAR**.
3. Fill in the fields as necessary. Ensure that the **.java** check box is selected.
4. Ensure that you select your database.
5. Click **Finish**.

Importing your enterprise beans into WebSphere Studio Application Developer

1. In WebSphere Studio Application Developer, create a new EJB 1.1 project and import your bean, by doing the following:
   a. Switch to the J2EE Hierarchy view and expand **Enterprise Applications**.
   b. Right-click the **WebSphereCommerceServer** enterprise application and select **Import**.
      The Import wizard opens.
   c. Select **EJB JAR file** and click **Next**.
   d. In the **EJB JAR file** field, enter the name of the JAR file you created in in Exporting your enterprise beans from VisualAge for Java.
   e. Create a new EJB project by selecting **New** and then in the **New project name** field, enter a name for your new project. For example, enter `CustomizedDataEJBProject`. This project is referred to as *CustomizedDataEJBProject*.
   f. Ensure that the WebSphereCommerceServer enterprise application is selected as the existing application to which this new EJB project will be added.
   g. Click **Finish**.
   h. When prompted, confirm that you want to repair the server configuration. This makes the new project available when you run the WebSphereCommerceServer test server.
2. You must update the Java build path for your EJB project, as follows:
   a. Right-click *CustomizedDataEJBProject* and select **Properties**.
   b. In the left navigation pane, select **Java Build Path**.
   c. Click the **Projects** tab. Select any project that contains code that you reference. In particular, ensure that you select **wcs.ccore.ejbs**. Click **OK** then close the properties editor.
3. Right-click *CustomizedDataEJBProject* and select **Rebuild Project**.
4. If you have any errors (they will be listed in the Tasks view), refer to "Troubleshooting tips for migrating enterprise beans to WebSphere Studio Application Developer" on page 40 to troubleshoot them. After you have imported your enterprise beans, follow the instructions in "Locating enterprise bean information" on page 40 to determine where to find your enterprise bean and method properties, your schemas, maps, and so on.

Generating deploy code and access beans

1. In WebSphere Studio Application Developer, go to the J2EE Hierarchy view, expand the EJB Modules folder, and select the newly imported EJB JAR file.

2. From the **EJB JAR** pop-up menu, click **Generate** > **Deploy and RMIC Code**. Generate code for all your enterprise beans.
3. Right-click the EJB JAR file again and select **Generate** > **Access Beans** > **Edit Access Beans**.
4. Select **All Access Beans** and click **Finish**.
5. Right-click *CustomizedDataEJBProject* and select **Rebuild project**.

Troubleshooting tips for migrating enterprise beans to WebSphere Studio Application Developer: The following are some troubleshooting tips that may help when migrating enterprise beans to WebSphere Studio Application Developer:

- When migrating the method finder helpers, the finder helper interfaces will disappear, as they have moved into an XML description file. This will generate a problem since your finder helper object usually implements this interface; the implementation must be removed.
- If you use inheritance in your enterprise beans, and you have built your own custom finders, they may break, since the mapping of fields in the generated code is different in WebSphere Studio Application Developer. To fix this, go to the `EJSCMPxxxxxx` generated class, find the select statement generated for the `findbyPrimaryKey`, and use it as a template.
- If you set up the Preferences page within WebSphere Studio Application Developer, to stop an automatic build from being done every time you save changes, you must perform the following steps to generate the enterprise bean deployed code and RMIC stubs:
  1. Select your enterprise bean project, right-click and select **Generate** > **Deploy and RMIC code**.
  2. Since the RMIC compiler can only see compiled code and since the generated Java classes were not compiled yet, you will receive an error. After you receive this error, switch to Java perspective and build the project.
  3. Repeat step 1. You should not receive any errors this time.
  4. Repeat step 2 to compile your newly generated stubs so you do not receive run-time errors when you deploy your enterprise beans.

Locating enterprise bean information: The following table contains a list of enterprise bean items and where to find your enterprise beans after you have migrated them to WebSphere Studio Application Developer:

Table 1. Locating enterprise bean information after migrating to WebSphere Studio Application Developer, Version 5

Item	Location
Enterprise beans (fields, classes)	Expand the EJB Modules folder in the J2EE Hierarchy view.

Table 1. Locating enterprise bean information after migrating to WebSphere Studio Application Developer, Version 5 (continued)

Item	Location
Enterprise beans (finder classes, access beans, generated classes)	Expand the EJB Modules folder in the Navigator view and go to the `com` directory (or your package structure).
Associated information	Expand the EJB Modules folder in the J2EE Hierarchy view and select the enterprise bean JAR file you want to work with. From its pop-up menu, select **Open With** > **EJB Deployment Descriptor**. In the Deployment Descriptor, select the **Overview** tab.
Finder helper description	Expand the EJB Modules folder in the J2EE Hierarchy view and select the enterprise bean JAR file you want to work with. From its pop-up menu, select **Open With** > **EJB Deployment Descriptor**. In the Deployment Descriptor, select the **Beans** tab.
Mapping or schema description	Expand the EJB Modules folder in the J2EE Hierarchy view and select the EJB JAR file you want to work with. From its pop-up menu, select **Generate** > **EJB to RDB mapping**.
Transaction demarcation	Expand the EJB Modules folder in the J2EE Hierarchy view and select the enterprise bean JAR file you want to work with. From its pop-up menu, select **Open With** > **EJB Editor**. In the EJB Editor, select the **Beans** tab.
Isolation levels or find for update or read only methods marking	Expand the EJB Modules folder in the J2EE Hierarchy view and select the EJB JAR file you want to work with. From its pop-up menu, select **Open With** > **EJB Deployment Descriptor**. In the Deployment Descriptor, select the **Access** tab.
EJB environment variables	Expand the `EJB Modules` folder in the J2EE Hierarchy view and select the enterprise bean JAR file you want to work with. From its pop-up menu, select **Open With** > **EJB Editor**. In the EJB Editor, select the **Beans** tab.

Migrating EJB access beans: When enterprise beans are exported from VisualAge for Java Enterprise Edition, Version 4.0 using the Export Tool for Enterprise Java Beans 1.1, the metadata for any associated Java bean wrapper and copy-helper access beans will also be exported. Rowset access beans are not supported by WebSphere Studio Application Developer, so the metadata for these access beans will not be exported.

Migrating customer finder helpers: When you export enterprise beans from VisualAge for Java Enterprise Edition, Version 4.0 using the Export Tool for Enterprise JavaBeans 1.1, or when 1.0 JAR files are deployed with the Deployment Tool for Enterprise JavaBeans (that is, the EJB Deploy Tool), the finder helper interfaces are migrated to the extension document. If the JAR file is an enterprise bean JAR file, the metadata is also migrated to the extension document from the finder helper interfaces. However, migration of the finder helper interfaces to the extension document only occurs if the finder descriptors in the JAR file are not found in the extension document. If the enterprise beans are exported using the Export Tool for Enterprise Java Beans 1.1, the redundant classes are filtered from the exported JAR. If the enterprise beans are not exported using the Export Tool for Enterprise Java Beans 1.1 and are imported along with the redundant classes, the classes are simply ignored.

Migrating customized WebSphere Commerce public entity beans into WebSphere Studio Application Developer

After migrating your own enterprise beans into WebSphere Studio Application Developer, the next step in transitioning your code to WebSphere Studio Application Developer is to migrate customized WebSphere Commerce public entity beans into WebSphere Studio Application Developer. This task involves two subtasks: migrating code and metadata changes into existing WebSphere Commerce Server projects in WebSphere Studio Application Developer, and then generating deploy code and access beans.

Migrating code and metadata changes into existing WebSphere Commerce Server projects in WebSphere Studio Application Developer

To migrate code and metadata changes into existing WebSphere Commerce Server projects in WebSphere Studio Application Developer, do the following:

1. In WebSphere Studio Application Developer, select the WebSphere Commerce Server enterprise bean project to which you want to make changes.
2. Merge your Java code changes. To do this, refer to the WebSphere Studio Application Developer documentation on how to make Java code changes in an enterprise bean project.
3. Merge your table or column definitions. To do this, refer to the WebSphere Studio Application Developer documentation on how to modify existing database table definitions in an enterprise bean project.

4. Merge your container managed persistence (CMP) fields. Refer to the WebSphere Studio Application Developer documentation on how to add or modify CMP fields for an enterprise bean.
5. Merge finder methods. Refer to the WebSphere Studio Application Developer documentation on how to add or modify finder methods in the deployment descriptor for an enterprise bean project.
6. Repeat the preceding steps for each WebSphere Commerce Server enterprise bean project you want to change.

Note: You may encounter warnings or errors during the code merge process. This is normal and errors are fixed when you generate the deploy code and access beans, as described in Generating deploy code and access beans.

Generating deploy code and access beans

To regenerate the deploy code and access bean, do the following:

1. In WebSphere Studio Application Developer, open to the the J2EE perspective, and select the J2EE hierarchy. Expand **EJB Modules**.
2. Right-click the EJB project and select **Generate** > **Deploy and RMIC Code**.
3. Select **All EJBs** and click **Finish**.
4. Right-click the EJB project again and select **Generate** > **Access Beans** > **Regenerate Access Beans** .
5. Select **All Access Beans** and click **Finish**. Note that additional migrated fields may need to be selected before generating.
6. Right-click the EJB project and select **Rebuild project**.

Note: In addition, refer to the following information about migrating enterprise beans into WebSphere Studio Application Developer:

- "Troubleshooting tips for migrating enterprise beans to WebSphere Studio Application Developer" on page 40
- "Locating enterprise bean information" on page 40
- "Migrating EJB access beans" on page 42
- "Migrating customer finder helpers" on page 42

Setting up your test environment and testing your migrated application

After migrating your enterprise bean files into WebSphere Studio Application Developer, the last step in transitioning your code to WebSphere Studio Application Developer is to set up your test environment and test your migrated application.

To start the server instance, follow these steps:

1. Go to the Servers view. If you cannot find it in the perspective, on the main menu select **Perspective** > **Show View** > **Servers**.

2. Right-click the **WebSphereCommerceServer** test server and select **Publish**.
3. When the publishing is complete, close the pop-up menu by clicking **OK**.
4. Right-click the **WebSphereCommerceServer** test server and select **Start**. The Console view displays.
5. Follow the startup process for the instance in the console.

Next, you should develop and execute test cases to ensure that your customized code functions as designed.

Chapter 9. Code transition tutorials

In order to perform the steps in this chapter, you must have completed the tutorials from the *WebSphere Commerce Version 5.4 Programmer's Guide* within VisualAge for Java.

Tutorials for transitioning your code

The following tutorials illustrate how to effectively migrate customized or extended code. Each tutorial is an extension of the tutorials within the *WebSphere Commerce Version 5.4 Programmer's Guide*. As an example, the *Programmer's Guide* contains a tutorial that illustrates how to extend a controller command, and the migration tutorial contained in this document explains how to migrate that extended command into WebSphere Studio Application Developer.

Before you start the tutorials, ensure that you have migrated all the store assets from VisualAge for Java into your WebSphere Studio Application Developer development environment. The following is a list of steps to migrate your JSP files into WebSphere Studio Application Developer:

1. Open WebSphere Studio Application Developer and switch to the J2EE Navigator view.
2. Navigate to the **wcstores\Web content** directory, right-click this directory and select **Import**
3. Select **Folder** and click **Next**.
4. Browse to the directory *VAJ_installdir*\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web
5. Select all files to import, including sub-folders.
6. Click **Finish**.

Tutorial A: Migrating an extended or customized controller command

The following tutorial illustrates how to migrate an extended controller command. This migration involves several sub-steps: migrating the controller command itself, migrating the JSP file displaying the command, migrating the enterprise bean associated with the command, and testing that everything migrated successfully. This tutorial can act as a model for migrating other extended or customized controller commands.

Migrating an extended controller command

To migrate an extended controller command, you must first export the project to a JAR file. Next, you import the Java and resource files into WebSphere Studio Application Developer, and then compile the migrated code.

To export your `_WCSamples` project to a JAR file, follow these steps:

1. In the VisualAge for Java Workbench window, select **_WCSamples project**, right-click, and select **Export**.
2. Select the **Jar file** radio button and click **Next**.
3. Specify the name of the JAR file.
4. Select the **.java** check box to export your Java files.
5. Fill in the other fields as required. Refer to the VisualAge for Java online help for more information on how to perform this task.

To import your Java and resource files into WebSphere Studio Application Developer, follow these steps:

1. Open WebSphere Studio Application Developer and switch to the Resource perspective.
2. Select **File** > **Import** > **Zip file**. Click **Next**.
3. Browse to the appropriate JAR file created in the preceding section.
4. Select all .java files in the JAR to import.
5. For the folder to contain your source files, specify `wcsstores/Java Source`.

To compile migrated code in the Web project, follow these steps:

1. Select the **wcstores** project.
2. Right-click and select **Build Project**.

Migrating a customized JSP file

In the same *Programmer's Guide* tutorial, you modified the confirmation.jsp template to display new business logic that you added to the order process business logic. To migrate a customized JSP file to WebSphere Studio Application Developer, follow these steps:

1. Open WebSphere Studio Application Developer and switch to the Resource perspective.
2. Click **File** > **Import** > **Folder** and click **Next**.
3. Browse to the directory *VAJ_installdir*\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\storeDir
4. Select the confirmation.jsp file.
5. Specify the `wcstores/Web Content` folder to contain the file.

Migrating an enterprise bean

In the same *Programmer's Guide* tutorial, you created a BonusBean enterprise bean. Migrating this enterprise bean to WebSphere Studio Application Developer involves several sub-steps: exporting the enterprise bean as an EJB 1.1. JAR using the VisualAge for Java EJB Export Tool, importing the enterprise bean into WebSphere Studio Application Developer, and generating the deploy code and associated access bean.

Exporting your enterprise bean from VisualAge for Java:

1. Within VisualAge for Java 4.0, on the EJB page of the Workbench, right-click the **WCSSamplesEntityBean** enterprise bean group and click **Export** > **EJB 1.1 JAR**.
2. Fill in the fields as necessary. Ensure that the **.java** check box is selected.
3. Select **UDB DB2 V7.2** as your database.
4. Click **Finish**.

Importing your enterprise bean into WebSphere Studio Application Developer:

1. In WebSphere Studio Application Developer, create a new EJB 1.1 project and import your bean, by doing the following:
   a. Switch to the J2EE Hierarchy view and expand **Enterprise Applications**.
   b. Right-click the **WebSphereCommerceServer** enterprise application and select **Import**.
      The Import wizard opens.
   c. Select **EJB JAR file** and click **Next**.
   d. In the **EJB JAR file** field, enter the name of the JAR file containing the bean you exported from VisualAge for Java.
   e. Create a new EJB project by selecting **New** and then in the **New project name** field, enter a name for your new project. For example, enter `CustomizedDataEJBProject`.
   f. Ensure that the WebSphereCommerceServer enterprise application is selected as the existing application to which this new EJB project will be added.
   g. Click **Finish**.
   h. When prompted, confirm that you want to repair the server configuration. This makes the new project available when you run the WebSphereCommerceServer test server.
2. You must update the Java build path for your EJB project, as follows:
   a. Right-click *CustomizedDataEJBProject* and select **Properties**.
   b. In the left navigation pane, select **Java Build Path**.
   c. Click the **Projects** tab. Select any project that contains code that you reference. In particular, ensure that you select **wcs.ccore.ejbs**. Click **OK** then close the properties editor.
3. Right-click *CustomizedDataEJBProject* and select **Rebuild Project**.

Generating deploy code and an access bean:

1. In WebSphere Studio Application Developer, go to the the J2EE Hierarchy view, expand the EJB Modules folder,
2. Right-click *CustomizedDataEJBProject* and select **Generate** > **Deploy and RMIC Code**.
3. Select **All EJBs** and click **Finish**.
4. Right-click the *CustomizedDataEJBProject* again and select **Access Beans** > **Regenerate Access Beans**.
5. Select **All Access Beans** and click **Finish**.
6. Right-click *CustomizedDataEJBProject* and select **Rebuild project**.

Note: Since this newly created enterprise bean project is required for the `wcstores` Web project, you need to add this new project to the build path of the `wcstores` Web project. This can be done by selecting and right-clicking the `wcstores` project, selecting **Properties**, and then selecting **Java build path**.

Testing your new logic in the sample store

To test the migrated business logic, do the following:

1. In the Servers view, right-click **WebSphereCommerceServer** and select **Start**.
2. Once the WebSphere Commerce Server instance has started inside of WebSphere Studio Application Developer, open a browser and type the URL for your store's home page. For example, type the following URL:
   `http://localhost/webapp/wcs/stores/servlet/StoreCatalogDisplay?`
   `storeId=`*`store_Id`*`&catalogId=`*`catalog_Id`*`&langId=-1`
   where *store_Id* is the identifier for your store and *catalog_Id* is the identifier for your store's catalog.
3. Select and purchase a product.
4. After you have purchased a product, the order confirmation displays the number of bonus points that were earned on the order.

Tutorial B: Migrating an extended or customized entity bean and task command

The following illustrates how to migrate an extended entity bean and an extended task command. This migration involves several sub-steps: migrating the task command , the extended data bean, the enterprise bean associated with the command, the JSP file displaying the command, as well as testing that everything migrated successfully.

The "Modifying an existing entity bean and extending an existing task command"tutorial from the *Programmer's Guide* illustrates how to customize an existing task command, GetProductContractUnitPriceCmd, and its corresponding interface. It also illustrates how to modify a WebSphere Commerce public entity bean. As part of that tutorial, you created new business logic to calculate a discounted price, based upon the customer's balance of bonus points. This calculation was performed by a new task command. The newly extended command, GetNewProductContractUnitPriceCmd, and its interface, GetNewProductContractUnitPriceCmdImpl, will be used as a model for this migration tutorial.

Migrating an extended task command and data bean

To migrate an extended task command and data bean, you must export the project from VisualAge for Java to a JAR file, import the Java and resource files into WebSphere Studio Application Developer, and then compile the migrated code.

To export your `_WCSamples` project to a JAR file, follow these steps:

1. In the VisualAge for Java Workbench window, select **_WCSamples** project, right-click this project, and select **Export**.
2. Select the **Jar file** radio button and click **Next**.
3. Specify the name of the JAR file.
4. Select the **.java** check box to export your Java files.
5. Fill in the other fields as required. Refer to the VisualAge for Java online help for more information on how to perform this task.

To import your Java and resource files into WebSphere Studio Application Developer, follow these steps:

1. Open WebSphere Studio Application Developer and switch to the Resource perspective.
2. Select **File** > **Import** > **Zip file**. Click **Next**.
3. Browse to the appropriate JAR file created in the preceding section.
4. Select all .java files in the JAR to import.
5. Choose the `wcsstores/Java Source` folder to contain your files.

To compile migrated code in the Web project, follow these steps:

1. Select the **wcstores** project.
2. Right-click and select **Build Project**.

Migrating an enterprise bean

In the same *Programmer's Guide* tutorial, the User enterprise bean is customized in a manner such that it appears to applications that the BONUSPOINT column of the BONUS table is actually a column in the USERS table. When a new record is created in the USERS table, a corresponding record is automatically inserted into the BONUS table. In order to create this table join, a new container managed persistence (CMP) field must be added to the User entity bean. This CMP field maps to the BONUSPOINT column in the BONUS table using the Secondary Table Map feature.

Migrating this enterprise bean to WebSphere Studio Application Developer involves several sub-steps: migrating the CMP filed to the User entity bean, updating the USER database schema and table mapping to include the BONUS table, and generating the deploy code and access bean for the User entity bean.

Migrating the bonusPoint field to the User entity: In this section, you will use the enterprise bean tools to migrate the CMP field to the entity bean. The field is called bonusPoint and is eventually be mapped to the BONUSPOINT column of the BONUS table. To recreate the CMP field to the User entity bean, do the following:

1. In WebSphere Studio Application Developer, go to the J2EE perspective and from the J2EE hierarchy, expand **EJB Modules**.
2. Expand **User EJB module**, and then **Member**.
3. Right-click **User** and select **Open With** > **Deployment Descriptor Editor**.
4. Go to the Bean page and select **User EJB**.
5. On the right, click **Add** to bring up the CMP attribute wizard and provide the following information:
   - **Name**: bonusPoint
   - **Type**: int Initial
   - **Value**: 0
   - **Access with getter and setter methods**: enabled
   - **Promote getter and setter methods to remote interface**: disabled
6. Click **OK**.
7. Save your EJB Deployment Descriptor changes (**Ctrl+S**).

The bonusPoint field is displayed in the Attribute window. Note that warnings are displayed, but these are fixed when you regenerate the entity bean's code, as described in "Generating deploy code and the access bean" on page 51.

Updating the schema and table mapping to include the BONUS table: You can now update the User schema with the BONUS table, create the foreign key relationship for the new table, and create a table map between the fields of the User entity bean and the columns of the BONUS table.

To create the table schema, do the following:

1. In WebSphere Studio Application Developer, go to the J2EE perspective, and from the J2EE hierarchy, expand **Databases**.
2. Expand **User database schema**, and then **NULLID**.

3. Right-click **Tables** and select **New** > **New Table Definition**. The Table Definition Editor opens.
4. Type `BONUS` for the table name and click **Next**.
5. In the Table Columns page, create the following two columns:
   - memberId - BIGINT, key column
   - bonusPoint - INTEGER, nullable
6. Click **Next** twice to go to the Foreign Keys page.
7. Click **Add Another** to create a new foreign key with the following information:
   - **Foreign key name**: F_User_Bonus
   - **On Delete**: CASCADE
   - **Target table**: NULLID.USERS
8. From **Source Columns**, select **member_Id** and click the **>** button.
9. Click **Finish** to create the new table.

To create the BONUS table map, do the following:
1. In WebSphere Studio Application Developer, go to the J2EE perspective, and from the J2EE hierarchy, expand **EJB Modules**.
2. Expand **User EJB module**, and then **Maps**.
3. Right-click **User map** and select **Open With** > **Mapping Editor**.
4. From the upper right corner pane, hold down the **Ctrl** key and select the **USERS** and **BONUS** tables. Both tables should be highlighted.
5. From the pull-down menu, choose **Mapping** > **Create Mapping**.
6. From the pull-down menu, choose **Mapping** > **Match by Name**.
7. Save your EJB mapping (**Ctrl+S**).

At this point, the User bean contains errors indicating that some abstract classes are not implemented. These errors are fixed when deployed code is generated.

Generating deploy code and the access bean: Since you have modified the code for the User entity bean, you must regenerate its deployed code and its access bean.

To regenerate the deployed code, do the following:
1. In WebSphere Studio Application Developer, go to the the J2EE perspective, and from the J2EE hierarchy, expand **EJB Modules**.
2. Right-click the **User EJB** module and select **Generate** > **Deploy and RMIC Code**.
3. Select **All EJBs** and click **Finish**.

4. Right-click the **User EJB** module and select **Generate** > **Access Beans** > **Edit Access Beans**.
5. Select **All Access Beans** and click **Finish**.
6. Right-click the EJB project and select **Rebuild project**

Migrating a customized JSP file

In the same *Programmer's Guide* tutorial, you modified the ProductDisplay.jsp template so that customers will be able to see the customized price that you added to the order process business logic. To migrate a customized JSP file to WebSphere Studio Application Developer, follow these steps:

1. Open WebSphere Studio Application Developer and switch to the Resource perspective.
2. Click **File** > **Import** > **Folder** and click **Next**.
3. Browse to the directory *VAJ_installdir*\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\store_directory
4. Select ProductDisplay.jsp to import.
5. Specify `wcstores/Web Content` as the folder to contain your files.

Testing your new logic in the sample store

To test the migrated code, do the following:

1. From the Servers view, right-click **WebSphereCommerceServer** and select **Start**.
2. Once the WebSphere Commerce Server instance has started inside of WebSphere Studio Application Developer, open a browser and type the URL for your store's home page. For example, type the following URL:
   `http://localhost/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=`*`store_Id`*`&catalogId=`*`catalog_Id`*`&langId=-1`
   where *store_Id* is the identifier for your store and *catalog_Id* is the identifier for your store's catalog.
3. Click the **Register** link under the **Services** heading and then click **Register** under the **New Customer** heading.
4. Register a new customer using the e-mail address `wctester@wc` and the password `wctester1`.
5. Fill in other fields with test values and click **Submit** and leave the browser open.
6. Open the DB2 Command Center and do the following:
   a. Click the **Interactive** tab.
   b. In the **Command** field, do the following:
      1) Type:
      ```
      connect to your_database_name
      ```

where *your_database_name* is the name of your WebSphere Commerce database

2) Click the **Execute** icon.

3) Type:

```
select users_id from userreg where logonid = 'wctester@wc'
```

4) Click the **Execute** icon.

c. The **Query Results** tab displays the entry for the customer you registered in the previous step. Make note of the customer's USERS_ID value.

d. Update the newly registered customer's balance of bonus points.

e. Click the **Interactive** tab and in the **Command** field, type the following:

```
update BONUS set BONUSPOINT = 1000 where MEMBERID = users_id
```

where *users_id* is the value from 6c.

f. Click the **Execute** icon.

7. In the browser, click the **Men's** link to view the men's fashions section of the store.
8. Click the link for the featured special to view the product page. The page displays the regular price, and the discounted price based upon the customer's balance of bonus points.

Part 3. Code deployment

This part describes how to deploy customized WebSphere Commerce code from the WebSphere Studio Application Developer development environment to a target WebSphere Commerce Server.

Chapter 10. Deploying customized code

Overview of code deployment

When customizing your e-commerce application, you may do any of the following:

- Create new commands, data beans or entity beans
- Extend existing WebSphere Commerce entity beans
- Modify the logic of existing commands or data beans

When developing code within WebSphere Studio Application Developer, you can test your code within the WebSphere test environment. At some point, you must deploy your code to a WebSphere Commerce Server outside of the development environment. Note that the level of code on your *target WebSphere Commerce Server* must be the same as the level of code on the development machine.

In the following sections, *target WebSphere Commerce Server* refers to the WebSphere Commerce Server to which you are deploying the customized code. In some testing scenarios, you may deploy to a WebSphere Commerce Server that is running on the same machine as WebSphere Studio Application Developer. In other situations, the target WebSphere Commerce Server is on another machine, and may even be running on a different platform.

The following sections describe the high-level steps for deploying the various types of customized code. Use them to understand the steps involved in the deployment process but refer to "Appendix B Deployment details" of the *WebSphere Commerce Programmer's Guide* for step-by-step deployment instructions.

In addition to the following sections that describe deployment of customized code, if you have created new access control policies in your development environment, the same access control policies must be created on the target WebSphere Commerce Server. Refer to the "Loading the access control policies for the new resources" topic starting on page 272 in the tutorials contained in the *Programmer's Guide* for an example of this procedure.

All references to particular pages in the *Programmer's Guide* refer to the version of the document with the following version information in the Edition notices (inside the front cover):
Second Edition (June 2002, Revision 1 released in September 2002)

Alternative approach to deployment

In the future, an alternative and more automated approach to deployment of customized code will be provided. Check the WebSphere Commerce Web site in the future for more information.

Deployment of new commands and data beans

When creating new commands and data beans, you should place them in a package that is named appropriately for your application. For example, you could create a new package by the name of `com.mycompany.mycommands` in which you keep your new commands. This package must be stored within the stores web project, `wcstores`. Careful organization of code is required, to ensure that deployment is successful.

Note that the major difference when deploying new commands and data beans from WebSphere Studio Application Developer instead of from VisualAge for Java is the fact that a `.class` file is now used instead of a JAR file. As such, the steps that were previously described about repackaging JAR files are no longer applicable. Additionally, when you copy the `.class` file to the appropriate directory on the target machine, you must also maintain the correct directory structure to represent the package naming convention for the `.class` file.

With all of your new commands and data beans stored in the Java source directory of the wcstores Web project, deployment consists of the following high-level steps:

1. From the development machine, copy the `.class` file that contains the new command and data bean code into the following directory on the target WebSphere Commerce Server.

   ```
   drive:\WebSphere\AppServer\installedApps\
   WC_Enterprise_App_instanceName.ear\wcstores.war\WEB-INF\classes
   ```

   As an example, if you had `mycommand.class` class file, you might do the following:

   a. Locate the `mycommand.class` class file on your development machine:

      ```
      workspace_dir\myproject\bin\com\mycompany\mycomponent\mycommand.class
      ```

   b. Copy this class file into the following directory on the target WebSphere Commerce Server:

      ```
      drive:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_instanceName.ear\wcstores.war\
      WEB-INF\classes\com\mycompany\mycomponent\mycommand.class
      ```

2. Register the new command in the command registry on the target WebSphere Commerce Server. Refer to the “Command registration framework”section on page 26 of the *Programmer’s Guide* for more information.

3. Stop and restart the WebSphere Commerce enterprise application, using the WebSphere Application Server Administrator's Console. Refer to the appropriate *WebSphere Commerce Installation Guide* for more information about starting and stopping this application.

Deployment of new entity beans

When creating new entity beans, you must create them in your own EJB project that is separate from the EJB project that contains the WebSphere Commerce entity beans. You must also ensure that this project does not contain the code for any commands or data beans. For example, you could create a new EJB project called `MyEntityBeansProject`. If the project contains code other than the code for new entity beans, deployment may not be successful.

Once you are satisfied with the way your new entity beans function within the WebSphere test environment, you must deploy them to a target WebSphere Commerce Server.

In general, the deployment steps are very similar to those described in the *Programmer's Guide*, except for the following:

1. Instead of creating the EJB 1.1 JAR file, you now create a JAR file that includes the EJB deployed code. As such, you must follow the steps contained in this document to create the EJB JAR file, instead of the ones in the *Programmer's Guide* to create the EJB 1.1 JAR file.
2. Since the new EJB JAR file contains deployed code, you no longer have to perform the "Generating deployed code" section as described in the *Programmer's Guide*.
3. Creating the implementation JAR file for your own new entity beans is no longer required. As such, when you refer to the *Programmer's Guide*, ignore any steps related to the implementation JAR file for your own new entity beans.
4. Repackaging the JAR file, as described in the *Programmer's Guide* is no longer required.

The following information provides an overview of the deployment steps:

1. On the development machine, use WebSphere Studio Application Developer to create a new EJB JAR file for your new EJB project. To create this JAR file, do the following:
   a. In the J2EE perspective, highlight the EJB project that contains your new entity beans.
   b. From the **File** menu, select **Export**.
   c. Select **EJB JAR File** as the export destination and click **Next**.

d. From the **What resources do you want to export?** drop-down list, select the EJB project that you want to export.

e. Click **Browse** to select the directory into which the JAR file should be placed. In the **File name** field, enter a name for the JAR file and click **Open**. All of the resources in the Java output folder `bin` are included in the JAR file.

   Note: If this EJB project had been previously imported and the imported JAR file is specified on the class path, the contents of the specified JAR file will be merged back into the exported JAR. If a compiled class exists in the project, it will not be merged in from the imported JAR. As a result, if a Java class was deleted from the current project, during the export the class will be merged back into the new JAR file because it is contained in the `imported_classes` folder.

f. Specify additional options as required, using the **Options** check boxes.

g. Click **Finish**.

2. Copy the ***`EJBJarFile`***`.jar` JAR file into the following directory on the target WebSphere Commerce Server:

   `drive``:\WebSphere\CommerceServer\temp`
3. On the target WebSphere Commerce Server, modify the transaction isolation level of the enterprise beans that are contained in the JAR file. Use the `modifyIsolationLevel` command line utility provided with WebSphere Commerce to set the transaction isolation level to the appropriate level for your database type. Refer to the "Modifying transaction isolation level of entity beans" section on page 349 of the *Programmer's Guide* for step-by-step details.
4. On the target WebSphere Commerce Server, load the access control policies for any new resources that you have created. Use the `acpload` and `acpnlsload` commands to load the policy information. For an example of how to load access control policy information, refer to the "Loading the access control policies for the new resources" section of the *Programmer's Guide* for step-by-step details.
5. On the target WebSphere Commerce Server, export the current WebSphere Commerce enterprise application from the WebSphere Application Server. Once the export has completed, an `.ear` file is created that contains the entire application. Refer to the "Exporting the current WebSphere Commerce enterprise application"section on page 350 of the *Programmer's Guide* for step-by-step details.
6. On the target WebSphere Commerce Server, export the configuration information for the enterprise beans that are contained in the current WebSphere Commerce enterprise application that is running within WebSphere Application Server. The information is exported to an XML file by using the `-export` option of the XMLConfig command line utility

that is provided by WebSphere Application Server. You must add one new stanza to the generated XML file for each new entity bean that you have created. Refer to the "Exporting configuration information for enterprise beans"section on page 351 of the *Programmer's Guide* for step-by-step details.

7. Assemble your new enterprise bean or beans into the enterprise application using the Application Assembly Tool provided by WebSphere Application Server. Using this tool, you can open the `WC_Enterprise_App_`*`instanceName`*`.ear` file for the current application and then import any new enterprise beans. You can also specify configuration information such as class path information and security information using this tool. The end result of using this tool is that a new `.ear` file for the modified enterprise application is created. Refer to the "Assembling new enterprise beans into an enterprise application "section on page 357 of the *Programmer's Guide* for step-by-step details.
8. Stop and remove the current enterprise application from WebSphere Application Server. Refer to the "Stopping and removing an enterprise application"section on page 366 of the *Programmer's Guide* for step-by-step details.
9. Import your modified enterprise application (created in step 7) into WebSphere Application Server. Refer to the "Importing an enterprise application"section on page 366 of the *Programmer's Guide* for step-by-step details.

While attempting to run the XMLConfig -import command, you may receive the following error message: "Cannot expand the ear file under

Windows *`WAS_installdir`*`\installedApps\WC_Enterprise_App_`***`instanceName`***`.ear`

AIX Solaris *`WAS_installdir`*`/installedApps/WC_Enterprise_App_`***`instanceName`***`.ear`

400 `/QIBM/UserData/WebAsAdv4/`*`wasInstanceName`*`/installedApps/WC_Enterprise_App_`*`instanceName`*`.ear`
If you receive this message, remove or rename the preceding directory and run the command again.

---

10. Restart your WebSphere Commerce enterprise application. Refer to the "Starting an enterprise application"section on page 368 of the *Programmer's Guide* for step-by-step details.

Deploying extended commands and data beans

The method for extending existing commands depends upon the type of modification required. The methods for extension are explained in the “Customization of existing commands” section on page 139 of the *Programmer's Guide*.

In general, modification of existing WebSphere Commerce commands involves creating new classes that inherit from the classes that require customization. Override methods of the superclasses as required, to replace or modify logic. When customizing data beans, you also create a new class that extends an existing data bean. Within the new class, make the required modifications.

When creating these new classes, ensure that they are stored within one of your own packages, which is itself stored within the stores web project, `wcstores`.

Since extensions are effectively handled by subclassing, deployment for extensions to commands and data beans is the same as deployment for new commands and data beans. For more information about this type of deployment scenario, refer to “Deployment of new commands and data beans” on page 58.

Note that the major difference when deploying new commands and data beans from WebSphere Studio Application Developer instead of from VisualAge for Java is the fact that a `.class` file is now used instead of a JAR file. As such, the steps that were previously described about repackaging JAR files are no longer applicable.

Deploying modified WebSphere Commerce public entity beans

When you modify a WebSphere Commerce public enterprise bean you modify WebSphere Commerce code. The deployment technique for customized entity beans is, therefore, slightly different than that used for new entity beans.

In general, the deployment steps for modified WebSphere Commerce public enterprise beans are very similar to those described in the *Programmer's Guide*, except for the following:

1. Instead of creating the EJB 1.1 JAR file, you now create a JAR file that includes the EJB deployed code. As such, you must follow the steps contained in this document to create the EJB JAR file, instead of the ones in the *Programmer's Guide* to create the EJB 1.1 JAR file.
2. Since the new EJB JAR file contains deployed code, you no longer have to perform the "Generating deployed code" section as described in the *Programmer's Guide*.

3. Repackaging the JAR file, as described in the *Programmer's Guide* is no longer required.

The following provides an overview of the deployment steps:

1. On the development machine, use WebSphere Studio Application Developer to create a new EJB JAR file for your modified EJB project. To create this JAR file, do the following:
   a. In the J2EE perspective, highlight the EJB project that contains your modified entity beans. For example, if you had modified the User entity bean, highlight the **WCSUser** EJB project
   b. From the **File** menu, select **Export**.
   c. Select **EJB JAR File** as the export destination and click **Next**.
   d. From the **What resources do you want to export?** drop-down list, select the EJB project that you want to export.
   e. Click **Browse** to select the directory into which the JAR file should be placed. In the **File name** field, enter a name (for example, `Cust_WCSUser-ejb.jar`) for the JAR file and click **Open**. All of the resources in the Java output folder `bin` are included in the JAR file.

      Note: If this EJB project had been previously imported and the imported JAR file is specified on the class path, the contents of the specified JAR file will be merged back into the exported JAR. If a compiled class exists in the project, it will not be merged in from the imported JAR. As a result, if a Java class was deleted from the current project, during the export the class will be merged back into the new JAR file because it is contained in the `imported_classes` folder.
   f. Specify additional options as required, using the **Options** check boxes.
   g. Click **Finish**.
2. Copy the ***`Cust_EJBJarFile`***`.jar` JAR file into the following directory on the target WebSphere Commerce Server:

   `drive``:\WebSphere\CommerceServer\temp`
3. On the target WebSphere Commerce Server, modify the transaction isolation level of the enterprise beans that are contained in the JAR file. Use the `modifyIsolationLevel` command line utility provided with WebSphere Commerce to set the transaction isolation level to the appropriate level for your database type. Refer to the “Modifying transaction isolation level of entity beans” section on page 349 of the *Programmer's Guide* for step-by-step details.
4. On the target WebSphere Commerce Server, export the current WebSphere Commerce enterprise application from the WebSphere Application Server. Once the export has completed, an `.ear` file is created that contains the

entire application. Refer to the “Exporting the current WebSphere Commerce enterprise application”section on page 350 of the *Programmer's Guide* for step-by-step details.

5. On the target WebSphere Commerce Server, export the configuration information for the enterprise beans that are contained in the current WebSphere Commerce enterprise application that is running within WebSphere Application Server. The information is exported to an XML file by using the `-export` option of the XMLConfig command line utility that is provided by WebSphere Application Server. You must modify the stanza for each WebSphere Commerce public entity bean that you have modified to reflect the new JAR file that contains the modified code. Refer to the “Exporting configuration information for enterprise beans”section on page 351 of the *Programmer's Guide* for step-by-step details.
6. Assemble the modified WebSphere Commerce public entity bean or beans in the enterprise application using the Application Assembly Tool provided by WebSphere Application Server. Using this tool, you can open the `WC_Enterprise_App_`*`instanceName`*`.ear` file for the current application and perform a number of tasks. These include:
   a. Determining the class path information for the current version of the WebSphere Commerce public entity beans that you are replacing.
   b. Removing the current version of these entity beans.
   c. Importing the new version of these entity beans.
   d. Applying the previous class path information to the beans.
   e. Configuring WebSphere Application Server security for the methods contained in all of the enterprise beans in the modified EJB module.
   f. Saving the modification and creating a new `.ear` file for the modified enterprise application.

   Refer to the “Assembling modified enterprise beans into an enterprise application ”section on page 361 of the *Programmer's Guide* for step-by-step details.
7. Stop and remove the current enterprise application from WebSphere Application Server. Refer to the “Stopping and removing an enterprise application”section on page 366 of the *Programmer's Guide* for step-by-step details.
8. Import your modified enterprise application (created in step 6) into WebSphere Application Server. Refer to the “Importing an enterprise application”section on page 366 of the *Programmer's Guide* for step-by-step details.

While attempting to run the XMLConfig -import command, you may receive the following error message: "Cannot expand the ear file under

Windows `WAS_installdir\installedApps\WC_Enterprise_App_instanceName.ear`

AIX Solaris `WAS_installdir/installedApps/WC_Enterprise_App_instanceName.ear`

400 `/QIBM/UserData/WebAsAdv4/wasInstanceName/installedApps/WC_Enterprise_App_instanceName.ear`

If you receive this message, remove or rename the preceding directory and run the command again.

9. Restart your WebSphere Commerce enterprise application. Refer to the "Starting an enterprise application"section on page 368 of the *Programmer's Guide* for step-by-step details.

Part 4. Appendixes

Appendix. Hints and tips

Links to WebSphere Commerce tools

Once you start the WebSphereCommerceServer in WebSphere Studio Application Developer, you can use the following links to access WebSphere Commerce tools:

- For WebSphere Commerce Accelerator http://localhost/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=common.mcLogon
- For the Administration Console http://localhost/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=adminconsole.AdminConsoleLogon
- For Store Services http://localhost/webapp/wcs/tools/servlet/ToolsLogon?XMLFile=devtools.Logon

The initial user ID and password for the site administrator is "wcsadmin" (for both values).

Changing workspaces

When WebSphere Studio Application Developer is started for the first time, you are prompted to specify the workspace directory that you would like to use. At that time, you can specify that you would not like to be prompted for this information anymore. If you make this selection, but then later want to switch workspace directories, you can do the following:

1. At a command prompt, navigate to the *WSAD_installdir* directory.
2. If you would like to specify the workspace diretory that should be used for this particular time that the application is opened, enter the following command:
   ```
   wsappdev -data workspace_dir -showlocation
   ```
3. If you would like to be prompted for the workspace location, enter the following command:
   ```
   wsappdev -setworkspace
   ```

If you are not prompted, you can start the application and use the correct workspace, by doing the following:

Browser usage

Using Microsoft Internet Explorer Version 5.5

If you are using Microsoft Internet Explorer Version 5.5 to access WebSphere Commerce code that is running within the context of WebSphere Studio Application Developer, the browser may redirect to an error page, rather than returning the page that was requested. In order to prevent this behavior, you should apply the “324929: December, 2002 Cumulative Patch for Internet Explorer 5.5 Service Pack 2”.

To download the patch, visit the following Web site:

`http://www.microsoft.com`

Using Microsoft Internet Explorer Version 6.0

If you are using Microsoft Internet Explorer Version 6.0 to access WebSphere Commerce code that is running within the context of WebSphere Studio Application Developer, the browser may redirect to an error page when making SSL connections, rather than returning the page that was requested. In order to prevent this behavior, you should apply the “Internet Explorer 6 Service Pack 1”.

To download this service pack, visit the following Web site:

`http://www.microsoft.com`

Incorrect PATH environment variable value

If a NoClassDefFoundError occurs in the “main” thread when you start the WebSphereCommerceServer, there may be an incorrect forward slash or back slash at the end of your PATH environment variable.

If this occurs, do the following:

1. Right-click **My Computer** and select **Properties**.
2. Click the **Advanced** tab.
3. Click **Environment Variables**.
4. From the **User variables for** *userName* list, select **PATH** and click **Edit**. Ensure that there are no extra forward or back slashes at the end of the value.
5. From the **System variables** list, select **PATH** and click **Edit**. Ensure that there are no extra forward or back slashes at the end of the value.
6. If you made changes to the values, click **OK** until you close the window.

Disabling caching to view JSP template changes

You must perform two configuration steps in order to be able to view changes to JSP templates without restarting the WebSphereCommerceServer test server.

First, you must disable the WebSphere Commerce caching, by doing the following:

1. Outside of WebSphere Studio Application Developer, navigate to the following directory:
   enhancements_installdir\instances\instanceName\xml
2. Open the *instanceName*.xml file in a text editor.
3. Locate the following stanza and modify the `Enabled="true"` value to `Enabled="false"`, as shown:

   ```
   <Cache AutoPageInvalidation="True"
            CacheCleanupAgentHostname="piggy"
            CacheCleanupAgentPort="8080"
            CacheCleanupPollingInterval="600"
            CacheConnectionTimeout="120000"
            CacheDaemonBindAddress="localhost"
            CacheDaemonMaxThreads="64"
            CacheDaemonPort="16999"
            CacheDirsPerMember="100"
            CacheFilePath="D:\COMMER~1\instances\WCDEV\xml\cache"
            CacheStoreClassName="com.ibm.commerce.cache.FileSystemCacheStore"
            Enabled="false"
            MaxAllowedRefreshPeriod="3600"
            MaxObjectsPerMember="0"
            WCSAppPath="/webapp/wcs/stores/servlet"
            name="Caching SubSystem"
            useCacheCleanupTriggers="false">
   ```

   Note that other values from the preceding snippet are specific to the particular environment.

The second step you must perform is to enable reloading of JSP code within the wcstores Web project, as follows:

1. Open WebSphere Studio Application Developer.
2. In the J2EE Navigator, expand the **wcstores** project.
3. Double-click **Web Deployment Descriptor**.
4. Select the **Extensions** tab and enable **Reloading enabled**.
5. Set an appropriate value for the reload interval and save your changes.
6. Restart your WebSphereCommerceServer test server.

Determining which version of the enhancements is installed

To determine which version of the enhancements is installed on your development machine, do the following:

1. From the **Start** menu, select **Run**.
2. In the **Open** field, enter `regedt32` and click **OK**.
3. In the HKEY_CURRENT_USER on Local Machine window, expand the following folders: **SOFTWARE > IBM > WebSphere Commerce Studio Enhancements**.
4. The pane on the right displays information about the version of the enhancements that is installed. You can determine the following information:
   - The value for `Edition` indicates if you have installed the enhancements for WebSphere Commerce Studio, Business Developer Edition, or for WebSphere Commerce Studio, Professional Developer Edition. The values are “Business” and “Professional” respectively.
   - The value for `Modification` indicates if you have the fix pack 5.4.0.6 level of code, or the fix pack 5.4.0.6 *with* the Commerce Enhancement Pack (October 2002) level of code installed. The values are “Fix Pack” and “October Commerce Enhancement Pack” respectively.

Notices

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

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

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
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-chome, Minato-ku
Tokyo 106, 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.

Trademarks and service marks

Printed in USA