IBM WebSphere Commerce Studio

Migration Guide for WebSphere Commerce Studio

Version 5.5

IBM WebSphere Commerce Studio

Migration Guide for WebSphere Commerce Studio

Version 5.5

Note:
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 87.

First Edition, Second Revision (September 2004)

This edition applies to IBM WebSphere Commerce Studio Version 5.5 for Windows 2000 (product number 5724-A18) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

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

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

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

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

Contents

Preface . . . . **v**
Updates to this book v
Conventions used in this book vi
Where to find more information vii

Part 1. Pre-migration 1

Chapter 1. Prerequisites for migration . . . 3
Prerequisite hardware 3
Prerequisite software 4
  Upgrading Oracle databases 4

Chapter 2. Migration environments 7
Environment A — Pure development environment 9
Environment B — Development environment and WebSphere Commerce coexist but do not share assets 9

Chapter 3. Code changes affecting your migration 11
Version 5.1 specific code changes 11
  WebSphere Commerce Suite 5.1 and WebSphere Commerce Studio 5.1 setup environment 11
  Access control 11
  Custom enterprise beans 16
  Logon command 16
  Pricing 17
  Product Advisor 17
  Rule server administration commands . . 18
Version 5.4 specific code changes 23
  WebSphere Commerce 5.4 and WebSphere Commerce Studio 5.4 setup environment . 23
  Collaborative workspaces 23
  Request for Quotation (RFQ) 28
Database schema differences 28
API differences 28
Migrating enterprise beans from 1.0 to 1.1 specification 29
Checking the unique index in the MSGTYPES table 31
WebSphere Application Command Framework 32
J2EE Connector Architecture 32

Part 2. Migrating your WebSphere Commerce development environment 33

Chapter 4. Migrating your WebSphere Commerce development environment . . 35
Backing up WebSphere Commerce and WebSphere Commerce development environment assets 35
Upgrading your DB2 product level 35
Upgrading your Oracle product level . . . 36
Upgrading your WebSphere Commerce development environment 36
  Exporting task commands, controller commands, and data beans to a JAR file . 37
  Exporting enterprise beans to a JAR file. . 38
Applying WebSphere Commerce fix pack 2 . 40
Migrating the WebSphere Commerce development environment instance 40
Migrating the WebSphere Commerce development environment database schema . 44
Transitioning your customized or extended code 45
  Differences between VisualAge for Java and WebSphere Studio Application Developer. 45
  Transitioning extended or modified WebSphere Commerce business logic and code 47
Migrating the payment instance and database 59

Part 3. Post-migration 61

Chapter 5. Post-migration actions 63
Environment B post-migration 64

Part 4. Tutorials for transitioning your code 67

Chapter 6. Migration flow for the tutorials 69

Chapter 7. Tutorial A: Migrating an extended or customized controller command 71
Migrating an enterprise bean 71
Exporting your enterprise bean from VisualAge for Java 71
Deleting the ExtensionJDBCHelperBean in the WebSphereCommerceServerExtensionData folder 72
Importing your enterprise bean into WebSphere Studio Application Developer . 72
Editing methods to avoid java.rmi.RemoteException errors 72
Changing the access isolation level . . . 73
Generating deploy code 73
Migrating a customized JSP file 73
Migrating an extended controller command 74
Testing your new logic in the FashionFlow sample store 75

Chapter 8. Tutorial B: Migrating an extended or customized entity bean and task command. 77
Migrating an extended task command . . . 77
Migrating an extended data bean 78
Migrating an enterprise bean 79
Migrating the bonusPoint field to the User entity 79
Updating the schema and table mapping to include the BONUS table. 80
Generating deploy code 81
Migrating a customized JSP file 81
Testing your new logic in the FashionFlow sample store 81

Part 5. Appendixes 85

Notices 87
Trademarks 89

Preface

This book describes how to migrate your existing version of the WebSphere® Commerce development environment (that is, WebSphere Commerce Studio) to WebSphere Commerce Studio, Version 5.5.

If you have applied any of the fix packs or Commerce Enhancement Packs for your WebSphere Commerce development environment, migration from those code levels are also supported.

If you are installing WebSphere Commerce Studio, Version 5.5 on a machine that does not have earlier versions of the development environment and supported products installed, see the *WebSphere Commerce Studio Installation Guide* for your database.

Updates to this book

This migration guide, and any updated versions of this migration guide, will be available in the WebSphere Commerce Technical Library.

To learn about last-minute changes to the product, see the updated product README file, also available from the above Web site.

Any updates to this book will be summarized in this section.

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

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

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

Change	Chapter, section, or page affected
Updated the instructions to export multiple EJB groups.	See "If you have many custom enterprise beans" on page 38.

Conventions 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, icons, or menu choices.
- `Monospace type` indicates examples of text you enter exactly as shown, file names, and directory paths and names.
- *Italic type* is used to emphasize words. Italics also indicate names for which you must substitute the appropriate values for your system. When you see any of the following names, substitute your system value as described:

  drive The letter representing the drive on which you installed the product or component being discussed (for example, C:).

  host_name
  The fully qualified host name of your WebSphere Commerce development environment machine (for example, ibm.com is fully qualified).

  VAJ_installdir
  The installation path for VisualAge® for Java™. The default installation path is \VisualAge for Java. For example, c:\Visual Age for Java.

  WC_installdir
  The installation path for WebSphere Commerce 5.5. The default installation path is \WebSphere\CommerceStudio55\Commerce. For example, c:\WebSphere\CommerceStudio55\Commerce.

  WCDE_installdir
  The installation path for your WebSphere Commerce development environment (that is, for WebSphere Commerce Studio, Version 5.5). The default installation path is \WebSphere\CommerceStudio5. For example, c:\WebSphere\CommerceStudio5.

  WSAD_installdir
  The installation path for WebSphere Studio Application Developer, Version 5. The default installation path is WebSphere\Studio. For example, c:\WebSphere\Studio.

  WSAD_workspace
  The installation path for your WebSphere Studio Application Developer, Version 5 workspace. The default installation path is \WebSphere\workspace_db2. For example, c:\WebSphere\workspace_db2.

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

Business indicates information specific to the WebSphere Commerce development environment for WebSphere Commerce Business Edition.

Express indicates information specific to WebSphere Commerce - Express Developer Edition.

Professional indicates information specific to the WebSphere Commerce development environment for WebSphere Commerce Professional Edition.

Where to find more information

For information related to the WebSphere Commerce development environment, refer to the WebSphere Commerce development environment site.

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

- WebSphere Commerce Technical Library, which includes a section on downloading updated documentation, such as guides and the online help.
- WebSphere Commerce Support
- Product overview sites:
  - WebSphere Commerce Business Edition
  - WebSphere Commerce Professional Edition
  - WebSphere Commerce - Express

Part 1. Pre-migration

This part discusses important information you should review before you migrate your WebSphere Commerce development environment. Information about hardware and prerequisites, possible migration environments, and important code changes are included in this section.

Chapter 1. Prerequisites for migration

To use the migration instructions in this guide, your WebSphere Commerce development environment machine must meet the hardware and software requirements detailed in the following sections.

Note: You have likely already met the following hardware and software requirements since your WebSphere Commerce development environment is already at the Version 5.5 level. Use the information in this chapter if you need to refer to the requirements.

Prerequisite hardware

You require a dedicated Pentium® III 733 MHz IBM-compatible personal computer with the following:

- A minimum of 1.5 GB of random access memory (RAM).
- The following amounts of disk space for each option that you choose to install:
  - For store front asset development:
    - WebSphere Studio — 300 MB
    - Store Archive Tools — 25 MB
    - Blaze Advisor and Blaze Innovator Workbench — 55 MB
  - For back-end development:
    - WebSphere Application Developer for Java Enterprise — 1.2 GB
    - IBM® DB2® Universal Database — 300 MB
    - WebSphere Commerce development environment — 300 MB
    - Blaze Advisor™ Server — 45 MB
  - IBM Distributed Debugger — 160 MB
  - Applet Designer — 15 MB
  - Page Detailer — 10 MB

  You will also need an additional 500 MB on the C: drive. If your machine is formatted with FAT partitioning and the partition is over 1.024 GB, you will need twice as much free disk space. The installation will check for adequate free disk space and will warn you if there is not enough space.
- A CD-ROM drive.
- A graphics-capable monitor with a color depth of at least 256 colors, with screen resolution set to 800x600 or higher.
- A mouse or other pointing device.

- A local area network (LAN) adapter that is supported by the TCP/IP protocol.

Prerequisite software

Ensure that you meet the following minimum software requirements before migrating your WebSphere Commerce development environment:

- Windows® 2000 Server Edition, or Advanced Server Edition installed, with Service Pack 3 applied. You can obtain updates from http://www.microsoft.com.

Important

If you are using Windows NT® with your previous version of your WebSphere Commerce development environment, you must upgrade your operating system to Windows 2000 before migrating. This version of the WebSphere Commerce development environment is not supported on Windows NT.

- Internet Explorer 5.5 Service Pack 1 with the latest critical security updates, on each machine that you will use to access the WebSphere Commerce development environment. A copy of Microsoft Internet Explorer is available by way of download on the Microsoft® Web site at http://www.microsoft.com.
- After you upgrade your WebSphere Commerce development environment to the 5.5 level, you need to install the WebSphere Commerce fix pack 2, so that you are at the 5.5.0.2 level. This fix pack includes fixes and enhancements for the migration process. This fix pack is available from the WebSphere Commerce Support site. Click on the link for the edition you are using and see the section Software downloads. Download this fix pack, and follow the instructions that are provided to install it. (If you plan to upgrade from WebSphere Studio Application Developer, Version 5 to WebSphere Studio Application Developer, Version 5.1, instructions are also provided in the fix pack documentation.) You should not proceed with the migration steps in this guide (such as migrating your instance or database) without installing this fix pack.

Upgrading Oracle databases

If you used Oracle with WebSphere Commerce Studio, Version 5.1 or WebSphere Commerce Studio, Version 5.4, before migrating to WebSphere Commerce Studio, Version 5.5, you must upgrade Oracle 8.1.6 (for WebSphere Commerce Version 5.1) or Oracle 8.1.7 (for WebSphere Commerce 5.4) to Oracle 9i for WebSphere Commerce 5.5. To upgrade to version Oracle 9i, consult the *Oracle9i Database Migration Guide* available at the Oracle Migration Center (`http://otn.oracle.com/tech/migration/content.html`).

Note that direct migration to Oracle 9i from version 8.1.6 is not supported. To migrate to Oracle 9i, you need to upgrade to Oracle 8.1.7 first, following the instructions in Chapter 3: ″Upgrading a Database to the New Oracle9i Release″ in the *Oracle9i Database Migration Guide*. You can then follow this guide to migrate to Oracle 9i.

Chapter 2. Migration environments

When you migrate your WebSphere Commerce development environment, you can select to migrate to an environment that is comparable to that if you completed a new installation of your WebSphere Commerce development environment on Version 5.5. That is, you can remove the unneeded WebSphere Commerce assets from your machine, after the WebSphere Commerce development environment migration is complete.

You are not required to remove these assets if you wish to continue having a local instance of WebSphere Commerce. In this case, it is recommended that you migrate your local instance of WebSphere Commerce to the WebSphere Commerce 5.5 level.

The following environments are supported as the end result of a WebSphere Commerce development environment migration:

- Environment A — Pure development environment
- Environment B — Development environment and WebSphere Commerce coexist but do not share assets.

The migration paths for these scenarios are described in the following sections. In each case, the environments can use either a remote WebSphere Commerce database, or a local (stand-along configuration) database. In addition, each environment can be used for developing storefront assets (such as the look and feel of a store), store back office logic (such as custom commands and enterprise beans), or a combination of both storefront assets and back office logic.

The high-level steps to migrate to your WebSphere Commerce development environment using either environment A or B are as follows:

__ 1. Ensure you have reviewed and understood the information described in Chapter 1, “Prerequisites for migration,” on page 3 and Chapter 2, “Migration environments.”

__ 2. If you are migrating from WebSphere Commerce Studio, Version 5.1, ensure that you have a working VisualAge for Java 3.5.3 WebSphere Test Environment before you begin the migration. This ensures you have all required JSP files, commands, data beans, and enterprise beans before you begin migration.

__ 3. Back up your existing WebSphere Commerce development environment and WebSphere Commerce assets as outlined in “Backing up WebSphere Commerce and WebSphere Commerce development environment assets” on page 35.

__ 4. If you use DB2 as your database management system, upgrade to DB2 Version 8.1 as outlined in “Upgrading your DB2 product level” on page 35.

__ 5. If you use Oracle as your database management system, upgrade to to Oracle9i Database Release 2 (which is version 9.2.0.1.0) as outlined in “Upgrading your Oracle product level” on page 36.

__ 6. Upgrade your current WebSphere Commerce development environment software level to the Version 5.5 level. The detailed installation and upgrade steps are described in “Upgrading your WebSphere Commerce development environment” on page 36. This includes an upgrade to WebSphere Studio Application Developer.

__ 7. Install the IBM WebSphere Commerce 5.5.0.2 fix pack to your WebSphere Commerce development environment machine by following the steps in the ″Applying the fix pack to WebSphere Commerce Studio″ section of the guide provided with the fix pack.

This fix pack includes fixes and enhancements for the migration process. This fix pack is available from the WebSphere Commerce Support site. Click on the link for the edition you are using and see the section Software downloads. Download this fix pack, and follow the instructions that are provided to install it. You should not proceed with the migration steps in this guide (such as migrating your instance or database) without installing this fix pack.

__ 8. Migrate the existing WebSphere Commerce development environment instance to the Version 5.5 level as described in “Migrating the WebSphere Commerce development environment instance” on page 40.

Note: Instance migration involves automatically copying the store front or Web assets (that is, JSP files, HTML pages, or graphics) from VisualAge for Java to WebSphere Studio Application Developer. However, you should also ensure that you have accounted for all the assets created for your store front within your previous version of your WebSphere Commerce development environment.

__ 9. Migrate the existing WebSphere Commerce development environment database schema as described in “Migrating the WebSphere Commerce development environment database schema” on page 44.

__ 10. Transition your WebSphere Commerce customized code contained in VisualAge for Java to WebSphere Studio Application Developer, Version 5. The detailed steps are described in "Transitioning your customized or extended code" on page 45.

__ 11. Migrate the WebSphere Commerce payment database and instance as outlined in "Migrating the payment instance and database" on page 59. For WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.4, the payment database was required to be installed on a different machine than the one containing WebSphere Commerce. For WebSphere Commerce 5.5, you have the option of installing the payment database locally on the same machine as WebSphere Commerce Studio, Version 5.5.

__ 12. Complete the post-migration steps outlined in Chapter 5, "Post-migration actions," on page 63. This includes uninstalling some components of WebSphere Commerce.

Environment A — Pure development environment

In the case of a migration on a pure development environment, the WebSphere Commerce development environment is used for developing store assets and logic; it is not used to host a production store. The end result of such a migration is a similar structure to a brand new installation of the WebSphere Commerce development environment. That is, in a pure development environment, WebSphere Commerce does not exist on your machine after the migration and only the WebSphere Commerce development environment exists on the machine. In this case, your existing development store is migrated, and the FashionFlow sample store is installed.

Environment B — Development environment and WebSphere Commerce coexist but do not share assets

In the case of a development environment and WebSphere Commerce coexisting, but not sharing assets, the end result should be one machine housing a development environment that is completely separated from the local instance of WebSphere Commerce 5.5. That is, in this environment, WebSphere Commerce and the WebSphere Commerce development environment both exist on one machine. Both the development environment and WebSphere Commerce uses a database; the databases are identical in content, but are not shared. After migration is complete, each of the databases should continue to be the same, as should the XML files. Note, that you are not required to migrate the database twice. Instead, after migrating the WebSphere Commerce development environment, you should back up the already migrated database, then create a new database (for example, `Mall2`) and restore to the new database. For details on this, refer to step 4 on page 64.

Chapter 3. Code changes affecting your migration

Once you have understood the prerequistes for migration and the possible migration environments, you need to review the code changes from your WebSphere Commerce development environment that affect your migration. This chapter discusses the code changes you should note before migrating.

Version 5.1 specific code changes

The following information is specific to Version 5.1 customers. If you are migrating from WebSphere Commerce Studio, Version 5.1, you should note this section, in addition to the other sections within this chapter about code changes.

WebSphere Commerce Suite 5.1 and WebSphere Commerce Studio 5.1 setup environment

With WebSphere Commerce Studio 5.1, you were required to install WebSphere Commerce Suite 5.1 on your development machine. You were also required to create a WebSphere Commerce Suite 5.1 instance and publish a store before installing your WebSphere Commerce development environment. This was required so that the store assets would be available for the WebSphere Test Environment component of VisualAge for Java.

Access control

The following table describes the differences between WebSphere Commerce Suite 5.1 access control and WebSphere Commerce 5.5 access control. The main difference is that WebSphere Commerce Suite 5.1 uses programmatic resource level access control, while WebSphere Commerce 5.5 uses policy-based, resource level access control. In order to minimize the failure of any customized code, the WebSphere Commerce 5.5 runtime currently handles both the WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.5 access control related command methods. However, it is highly recommended that you migrate any customized code to use the WebSphere Commerce 5.5 methods, to make use of the policy-based access control model. Any given command should function properly if it is entirely using one of following the access control models:

- The recommended WebSphere Commerce 5.5 model uses the validateParameters and getResources methods.
- The WebSphere Commerce Suite 5.1 model uses the checkParameters, checkPermision, and getResourceOwners methods.

Table 1. Access control subsystem differences

Item	WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
Access control model	Role-Based Access Control In WebSphere Commerce Suite 5.1, command level access control is implemented using the ACCCMDGRP table. Resource level access control is done programmatically in your source code. Changes in resource level policies required that you recompile your source code.	Policy-Based Access Control In WebSphere Commerce 5.5, command level and resource level access control is implemented using the ACPOLICY table. You can change policies without recompiling your source code.
Databeans	Protected programmatically	Protected directly and indirectly using the Delegator interface. If the databean does not implement this interface, it can be populated by anyone. Furthermore, even when the databean implements the Delegator interface, if it returns null in the getDelgate method, it can also be populated by anyone.
getResources()	Not applicable	This command method is used to trigger resource level access control checking. It returns all the protectable primary resources accessed by this command. It returns null if no resources are being accessed by this command.
getResource Owners()	Default behavior: Returns the owner of the store if a valid store ID parameter is defined for the command, that is, from requestProperties or session. Returns EC_ACC_ALL_RESOURCES if no store ID is available, or if storeId is set to ECConstants.EC_NO_STOREID.	The getResourceOwners() method returns null by default. In order to simulate WebSphere Commerce 5.5 behavior, when performing command level access control check, (that is, performing an access check where the command is the protectable resource) the command framework will use the resource owners returned from getResourceOwners(). It will also use the resource owners as the owner of the command. For examples, see "Examples using getResourceOwners()" on page 15.

Table 1. Access control subsystem differences (continued)

Item	WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
checkPermission()	This method provides fine grain access control check. The WebSphere Commerce Suite 5.1 Command framework invokes this method before the performExecute() method. Returns true if the administration command does not have fine grain access control.	The checkPermission() method has been kept for backward compatibility but should no longer be used for access control.
checkParameters()	This is where WebSphere Commerce Suite 5.1 performs parameter checking. The default implementation does not result in any action. The performExecute() of the ControllerCommandImpl and TaskCommandImpl calls checkParameters(). Most commands call super.performExecute() as the first line in their performExecute() in order to invoke checkParameters().	This method is replaced by validateParameters() in WebSphere Commerce 5.5 to support the new access control model. The default implementation does not result in any action. For backward compatibility, the performExecute() of the ControllerCommandImpl and TaskCommandImpl calls checkParameters(). Most commands call super.performExecute() as the first line in their performExecute() for good programming convention. This method, checkParameters(), will be deprecated in the next release.

Targetable commands	The sequence of method calls is:
	`{`
	`Command.checkPermission();...`
	`  Command.performExecute();...`
	`  Command.checkParameters();`
	`}`
		If you want to migrate your commands to the WebSphere Commerce 5.5 access control model, you need to implement validateParameters(). If you used checkParameters() in Commerce Suite 5.1, move your logic to validateParameters() and remove checkParameters() from your code.
		
		The sequence of method calls is:
		`{`
		`Command.validateParameters();`
		
		`Command.getResources();`
		
		`Command.checkPermission();`
		`// for backward`
		`compatiblity only`
		
		`Command.performExecute();`
		
		`Command.checkParameters();`
		`// for backward`
		`compatiblity`
		`only`
		`}`

Table 1. Access control subsystem differences (continued)

Item	WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
Controller commands and views	In WebSphere Commerce Suite 5.1, any controller commands that are defined in the URLREG table, but do not have a corresponding entry in the ACCCMDGRP table, are not under access control. As such, they are accessible by all users, including guest shoppers. Similarly, views that are defined in the VIEWREG table, but do not have a corresponding entry in the ACCCMDGRP table, are also accessible by all users. **Note:** A controller command or view that is in the ACCCMDGRP table, and has MbrGrp_Id = -2 , (that is, assigned to Customer Access Group), is also accessible by all users.	In WebSphere Commerce 5.5, the access control model has changed. Now, if a controller command does not explicitly have an access control policy that grants all users access to the command, ordinary users cannot access the command — only site administrators. Similarly, an explicit access control policy that grants access to a view is required if a user accesses the view directly from a URL or a command redirects to the view.

Notes:

1. If you have added a controller command, and the new command:
   - Extends the WebSphere Commerce Suite 5.1 base implementation without specifying any new interfaces.

     In this case, no resource level policies need to be modified, since the new command is still implicitly implementing the base interface (the base interface is already specified in the policies).
   - Extends the WebSphere Commerce Suite 5.1 base implementation, but also implements its own new interface, and the base implementation returns resource-action pairs in its getResources() method.

     In this case, the new interface has to be included in a resource-level policy. If the new command is operating on the same resources as the base implementation, then the new action can simply be added to the action groups that contain the base interface action.

     WebSphere Commerce 5.5 will only add the command level policy for it during migration. If the WebSphere Commerce command implements getResources(), then you either have to determine what resources it is returning and create the appropriate resource level policy for your command, or, if you do not want resource level access control, you have to override getResources() on your command so that it returns a null value.

   To determine what the WebSphere Commerce 5.5 commands are returning for their getResources(), analyze the trace and look for `Action=`*`WCBECommand`* and find all the Protectable resources getResources() is checking for. In the trace above, the resource is Order.

For example, consider if after enabling the SERVER trace, you find the following in the logs:

```
[4/21/03 16:29:19:467 EDT] 5177d022 WC_SERVER d AccManager isAllowed isAllowed? User=-1000;
Action=com.fvt.ACCOrderItemAddCmd; Protectable=com.ibm.commerce.order.objects._Order_Stub;
Owner=7000000000000002000resource is Groupable

[4/21/03 16:29:19:467 EDT] 5177d022 WC_SERVER d AccManager isAllowed PASSED? =false
```

(Note that some of the above lines have been split across several lines for display purposes.)The meaning of the above trace is that the resource level policy is failing. In this case, ACCOrderItemAddCmd is extending from the Server OrderItemAdd command which implements getResources(). Therefore, by default, ACCOrderItemAdd also requires a resource level policy, unless the getResources() on it is changed to return null. This resource level policy is not added during migration, since it is not known what WebSphere Commerce 5.5 commands that you are extending.

In most cases, commands return access beans in the getResources() method. For example, returning com.ibm.commerce.xyz.objects.XYZAccessBean in getResources() will appear as com.ibm.commerce.xyz.objects._XYZ_Stub in the trace. This difference is becauseWebSphere Commerce 5.5 must narrow the access bean to its remote interface (since it is the remote interface of the EJB that actually extends the Protectable interface).

2. In WebSphere Commerce Suite 5.1, resource level access control was enforced programmatically, within the command logic. In WebSphere Commerce 5.5, resource level access control policies are specified externally, similar to the way command level access control policies are specified. During the migration, only command level access control policies are migrated from WebSphere Commerce Suite 5.1 to WebSphere Commerce 5.5. Any resource-level access control policies that are needed due to customization of the WebSphere Commerce Suite 5.1 default access control policies (which are stored in the ACCCMDGRP table), need to be added manually. Otherwise you will receive an unexpected access control violation exception.

Examples using getResourceOwners()

WebSphere Commerce Suite 5.1 commands:

- Commands that rely on default behavior.

  These commands do not implement getResourceOwners(), the default returns `null`.

  When performing command level access control checks, the command framework returns the store owner as the command owner. It will return the EC_SITE_ORGANIZATION when no store ID is available. EC_SITE_ORGANIZATION represents the Root Organization.
- Commands that implement the getResourceOwners() method

When performing the command level access check, the command framework performs an access check on the command for each resource owner returned by getResourceOwners(). For example, suppose that the getResourceOwners() method returns 2 owners: Organization 1 and Organization 2. The command framework will perform an access check on the command, first with Organization 1 as the owner. If this check passes, it will perform another access check on the same command, this time it will use Organization 2 as the command owner. It must pass both access checks.

New WebSphere Commerce 5.5 commands:

- These commands do not implement getResourceOwners(); the default returns null.
- When performing a command level access control check, the command framework will return the store owner as the command owner. It will return the EC_SITE_ORGANIZATION when no store ID is available.

Custom enterprise beans

You used VisualAge for Java 3.5.3 with WebSphere Commerce Studio 5.1 (for details see "VisualAge for Java 3.5.3 enterprise bean 1.0 JARs versus VisualAge for Java 4.0 enterprise beans 1.1 JARs" on page 46). If you have a minimal number of customer enterprise beans, it is recommended that you re-create the enterprise beans using WebSphere Studio Application Developer, Version 5. For details on how to do this, refer to the *WebSphere Studio Application Developer, Version 5 Migration Guide*.

If you have a large number of custom enterprise beans, you first need to export your enterprise beans from VisualAge for Java 3.5.3 to VisualAge for Java 4.0, then export to enterprise bean 1.1 JAR within VisualAge for Java 4.0, and finally import into WebSphere Studio Application Developer, Version 5. If you are a WebSphere Commerce Studio, Version 5.1 customer migrating your WebSphere Commerce development environment, contact WebSphere Commerce Support to obtain a copy of VisualAge for Java 4.0. It is recommended that you install VisualAge for Java 4.0 on a single machine as the transition machine for migrating enterprise beans.

Logon command

Note that there are three behavioral changes introduced to the Logon command in WebSphere Commerce 5.5:

- If any of the ancestral organizations to which a shopper or administrative user for a store belong are locked, they will not be able to log on to the store.
- If a user does not play a role in a store's organization or any of its ancestral organizations, WebSphere Commerce does not allow the user to log onto the store.

- If a users's registration approval status is pending approval, WebSphere Commerce does not allow that user to logon to the store.

This may impact users who are in pending approval state, since they could log on in WebSphere Commerce Suite 5.1 but can no longer do so in WebSphere Commerce 5.5.

Pricing

The following are the commands and methods in WebSphere Commerce 5.5 (introduced in WebSphere Commerce 5.4) that replace the WebSphere Commerce Suite 5.1 commands for pricing:

Task commands

- GetContractUnitPriceCmd replaces GetBaseUnitPriceCmd
- GetContractSpecialPriceCmd replaces GetBaseSpecialPriceCmd
- GetProductContractUnitPriceCmd replaces GetProductBaseUnitPriceCmd

Note: For backward compatibility, the WebSphere Commerce Suite 5.1 commands are maintained in WebSphere Commerce 5.5.

Data beans

For the following data beans, in WebSphere Commerce Suite 5.1, the method getCalculatedPrice() was available to retrieve the price. In WebSphere Commerce 5.5, it is replaced with a new method, getCalculatedContractPrice().

- ItemDataBean
- PackageDataBean
- ProductDataBean
- CatalogEntryDataBean
- InterestItemDataBean
- BundleDataBean

Note: For backward compatibility, the WebSphere Commerce Suite 5.1 methods will be maintained in WebSphere Commerce 5.5.

For more information on the above commands or method, see the WebSphere Commerce Production online help.

Product Advisor

The sample JSP files in WebSphere Commerce 5.5 (pe51.jsp, pc51.jsp, and sa51.jsp found in the *WC_installdir*\samples\web\pa directory) are the migrated versions of the files with the same names in Commerce Suite 5.1. For WebSphere Commerce 5.5, the datatype package names have changed as summarized in the table below. For any JSPs that reference these package names, you need to change occurrences of `com.ibm.commerce.datatype` to `com.ibm.commerce.pa.datatype` as summarized in the following table:

WebSphere Commerce Suite 5.1	**WebSphere Commerce 5.5**
com.ibm.commerce.datatype.DsString	com.ibm.commerce.**pa**.datatype.DsString
com.ibm.commerce.datatype.DsInteger	com.ibm.commerce.**pa**.datatype.DsInteger
com.ibm.commerce.datatype.DsDouble	com.ibm.commerce.**pa**.datatype.DsDouble
com.ibm.commerce.datatype.DsCurrency	com.ibm.commerce.**pa**.datatype.DsCurrency
com.ibm.commerce.datatype.DsDecimal	com.ibm.commerce.**pa**.datatype.DsDecimal
com.ibm.commerce.datatype.DsURLLink	com.ibm.commerce.**pa**.datatype.DsURLLink
com.ibm.commerce.datatype.DsImage	com.ibm.commerce.**pa**.datatype.DsImage
com.ibm.commerce.datatype.DsDate	com.ibm.commerce.**pa**.datatype.DsDate

Note that there is also a data type that was introduced in WebSphere Commerce 5.4, com.ibm.commerce.pa.datatype.DsLong, that you should use for `catentry_id` or other attributes with values larger than a typical integer.

Rule server administration commands

Rule service administration commands have changed both behavior and interface for WebSphere Commerce 5.5. The WebSphere Commerce Suite 5.1 versions of the commands, found in packages com.ibm.commerce.rules.commands and com.ibm.commerce.ruleservice.admin.commands, use the Scheduler to broadcast requests to all your application clones to add, change, remove or refresh rule services. For WebSphere Commerce 5.5, the commands have been replaced with better-named ones in the same packages. Also, the commands now operate in a *just-in-time* manner. For example, when you refresh a rule service, each application clone refreshes its own instance of that rule service as soon as it needs to execute the rule service again. This approach improves reliability and avoids unnecessary updates. If you have extended the rule service administration commands, you will need to examine the new commands to see how this change in behavior affects your customized extensions.

Table 2. Project and package level mapping of the rules system

Project name	**WebSphere Commerce Suite 5.1**	**WebSphere Commerce 5.5**
WCS Rules Deployment	com.ibm.commerce.rules.deployment.commands	
WCS Rules Project Support	com.ibm.commerce.rules.project	com.ibm.commerce.rules.project
		com.ibm.commerce.rules.beans[1]

Table 2. Project and package level mapping of the rules system (continued)

Project name	WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
WCS Rules System	com.ibm.commerce.rules	com.ibm.commerce.rules
	com.ibm.commerce.rules.commands	com.ibm.commerce.rules.commands
	com.ibm.commerce.rules.beans[1]	
	com.ibm.commerce.rules.blaze[2]	
	com.ibm.commerce.rules.messages	
	com.ibm.commerce.rules.selector	
	com.ibm.commerce.rules.selector.blaze	
	com.ibm.commerce.rules.util[3]	
		com.ibm.commerce.rules.ecmessages
		com.ibm.commerce.rules.exception
WCS Rules System Admin	com.ibm.commerce.ruleservice.admin.beans	com.ibm.commerce.ruleservice.admin.beans
	com.ibm.commerce.ruleservice.admin.commands	com.ibm.commerce.ruleservice.admin.commands
	com.ibm.commerce.ruleservice.admin.util	com.ibm.commerce.ruleservice.admin.util
		com.ibm.commerce.ruleservice.admin.beansrc
WCS Rules System Enterprise Beans	com.ibm.commerce.rules.objects	com.ibm.commerce.rules.objects
	com.ibm.commerce.rules.objimpl	com.ibm.commerce.rules.objimpl
	WCSMCRuleLayerEJBReserved	WCSMCRuleLayerEJBReserved
		WCSRuleServerEJBReserved
		com.ibm.commerce.rules.helpers
Rule Server Integration		com.ibm.commerce.rules.blaze [2]
		com.ibm.commerce.rules.blaze.exception
Rules Infrastructure		com.ibm.commerce.rules.repository
		com.ibm.commerce.rules.util[3]

Table 2. Project and package level mapping of the rules system (continued)

Project name	WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
Rules Infrastructure Common Services		com.ibm.commerce.services
		com.ibm.commerce.services. logging
Note: Even though a few packages are similar in WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.5, they may have completely different classes. Similar classes may or may not have similar methods. [1] [2] [3] These packages have moved to different projects in WebSphere Commerce 5.5.		

The tables in the sections below list the changes to the WebSphere Commerce Suite 5.1 controller commands and application programing interfaces for rule server administration.

Controller commands

The behavior of the rule server administration controller commands have changed. Generally in WebSphere Commerce 5.5, they update rule service configuration information in the database, rather than broadcast information to all application clones. An exception is com.ibm.commerce.ruleservice.admin.commands. BroadcastUpdateRuleServiceStatusCommand, which is used through corresponding URLs, and is not intended to be customized nor extended.

Refer to the WebSphere Commerce Development online help for more information on any of these commands.

In the following table, only the base names of the commands are listed, for brevity. The `com.ibm.commerce.ruleservice.admin.commands.` portion of the complete command name is not included. For example, the complete name for the AddRuleServiceCommand command is com.ibm.commerce.ruleservice.admin.commands. AddRuleServiceCommand.

Table 3. Rule server controller commands

WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
AddRuleServiceCommand	AddRuleServiceCommand
Did not exist	BroadcastUpdateRuleServiceStatusCommand
StopRuleServiceCommand	DisableRuleServiceCommand
EditRuleServiceCommand	EditRuleServiceCommand
StartRuleServiceCommand	EnableRuleServiceCommand

Table 3. Rule server controller commands (continued)

WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
RefreshRuleServiceCommand	RefreshRuleServiceCommand
RemoveRuleServiceCommand	RemoveRuleServiceCommand
CheckRuleServiceStatusCommand	UpdateRuleServiceStatusCommand

Application programming interface calls (task commands)

What used to be direct method calls in WebSphere Commerce Suite 5.1 are now task commands in WebSphere Commerce 5.5. The most commonly-used application programming interface (API) call in WebSphere Commerce Suite 5.1 is to invoke a rule service. In WebSphere Commerce 5.5, this is done by using com.ibm.commerce.rules.commands. InvokePersonalizationRuleServiceCommand.

Refer to the WebSphere Commerce Development online help for more information on any of these task commands.

In the following table, only the base names of the APIs and commands are listed, for brevity. For the WebSphere Commerce Suite 5.1 commands, the `com.ibm.commerce.rules.RulesSystem.` portion of the complete API name is not listed. For example, the complete name for changeServiceConfiguration() is com.ibm.commerce.rules.RulesSystem.changeServiceConfiguration(). Similarly, for the WebSphere Commerce 5.5 commands, the com.ibm.commerce.rules.commands. portion of the command name is not listed. For example, the complete name for ChangePersonalizationRuleServiceCommand is com.ibm.commerce.rules.commands. ChangePersonalizationRuleServiceCommand.

Table 4. Rule server API calls (task commands)

WebSphere Commerce Suite 5.1	WebSphere Commerce 5.5
changeServiceConfiguration()	ChangePersonalizationRuleServiceCommand
addService()	CreatePersonalizationRuleServiceCommand
stopService()	DisablePersonalizationRuleServiceCommand
startService()	EnablePersonalizationRuleServiceCommand
invokeService()	InvokePersonalizationRuleServiceCommand
Did not exist	MarkPersonalizationRuleServiceChangedCommand
removeService()	RemovePersonalizationRuleServiceCommand
getService().getStatus()	UpdatePersonalizationRuleServiceStatusCommand

The following example illustrates implementation differences in API calls between WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.5:

Invoking a rule service in WebSphere Commerce Suite 5.1 (assuming the command context is set to `context` and the rule service name is `ruleServiceName`):

```
RuleServiceKey key=null;
key = new RuleServiceKey(ruleServiceName, context.getStoreId());
RulesSystem rulesSystem = RulesSystemToolbox.getInstance().getRulesSystem();
if (rulesSystem.isAvailable()) {
        rulesSystem.invokeService (key,context);
}
```

Invoking a rule service in WebSphere Commerce 5.5 (assuming the commandContext has been set to `commandContext` and the rule service name is `ruleServiceName`):

```
InvokePersonalizationRuleServiceCommand command =
    (InvokePersonalizationRuleServiceCommand)
CommandFactory.createCommand
    (InvokePersonalizationRuleServiceCommand.class.getName(), storeId);
command.setCommandContext(commandContext);
command.setServiceName(ruleServiceName);
command.execute();
```

Note that in the above example, the interface in WebSphere Commerce 5.5 is more straightforward and simpler to use. It hides many of the implementation details from a user.

Exception handling

In WebSphere Commerce 5.5, the rules system has extended exception handling to explicitly identify various problem areas. In WebSphere Commerce Suite 5.1, there was one `RulesExceptionHandler` class which handled all the generic rules system related exceptions. n WebSphere Commerce 5.5, the RulesExceptionHandler class has been replaced by a number of rules exception classes. These distinct classes make it easier to identify the actual cause of the exception. These classes are contained in the com.ibm.commerce.rules.exception package in the WebSphere Commerce rules system project and they extend the general WebSphere Commerce RuntimeException class.

The rules system exceptions are:

- InvalidRuleServiceKeyException
- PersonalizationRuleServerException
- PersonalizationRuleServerNotAvailableException
- PersonalizationRuleServiceExistsException
- PersonalizationRuleServiceNotEnabledException

- PersonalizationRuleServiceNotFoundException
- RuleServerConfigurationNotFoundException
- RuleServerNotFoundException
- RuleServiceNotFoundException
- RuleServiceConfigurationNotFoundException
- RulesSystemDataModelException
- RulesSystemRuntimeException

Version 5.4 specific code changes

The following information is specific to Version 5.4 customers. If you are migrating from WebSphere Commerce Studio, Version 5.4, you should note this section, in addition to the other sections within this chapter about code changes.

WebSphere Commerce 5.4 and WebSphere Commerce Studio 5.4 setup environment

With WebSphere Commerce Studio, Version 5.4, you had the option of your WebSphere Commerce development environment and WebSphere Commerce on the same machine so that they coexisted. You also had the option of installing a sample store that is used in the WebSphere Test Environment when you installed VisualAge for Java. If you selected the sample store, the following items were created:

- A database; for example, the default VAJ_DEMO database. You have the option to change this during the installation.
- The store directory structure.
- The instance XML file; for example, the default vaj_demo.xml instance XML file. You have the option to change this during the installation.

These assets are used by the WebSphere Test Environment. The available sample store for the WebSphere Commerce development environment was the InFashion store and theWebSphere Commerce development environment installation program used a master XML file to populate the sample store in the database.

Alternatively for your WebSphere Commerce development environment , you can maintain the WebSphere Commerce development environment and WebSphere Commerce on separate machines.

Collaborative workspaces

Business In WebSphere Commerce 5.4, the collaborative workspaces functionality were provided by the CollabManagerBean enterprise bean. In WebSphere Commerce 5.5, this enterprise bean has been replaced by the class CollabManager. If you have customized code that makes use of

CollabManagerBean enterprise bean, do the following to migrate your code to work with the new CollabManager class:

1. Edit the class or package importing statement to reference the new CollabManager class as follows:
   a. Remove the import statement in you code that refers to the CollabManagerBean class (for example, `import com.ibm.commerce.collaboration.objects.CollabManagerBean;`).
   b. Insert a new import statement that referse to the CollabManager class (for example, `import com.ibm.commerce.collaboration.manager.*;`).
2. Edit the declare and instance statements to use the CollabManager type.
3. Edit the methods that are invoked by the object according to the API mapping tables below. You may be required to add additional code depending on how you map the new API.

   Note: The parameter or return type of the equivalent API may not be the same as in CollabManagerBean. Check the CollabManager class API reference for detail.

   The following table illustrates the API mapping between CollabManagerBean class and CollabManager class (note that the the interfaces that are required by the enterprise bean specification are not listed):

Table 5. API mapping between CollabManagerBean class and CollabManager class

Method in CollabManagerBean	**Equivalent method in CollabManager class**
addAuthorGroupMember (CollabSpaceBean csbean)	Any of the following: • addCWMembers(CollabWorkspaceInfo csbean) • addCWMembers(String collabWorkspaceId,Vector memberDNs,String role) • addCWMember(String collabWorkspaceId,String memberDN,String role)
ddManagerGroupMember (CollabSpaceBean csbean)	Any of the following: • addCWMembers(CollabWorkspaceInfo csbean) • addCWMembers(String collabWorkspaceId,Vector memberDNs,String role) • addCWMember(String collabWorkspaceId,String memberDN,String role)

Table 5. API mapping between CollabManagerBean class and CollabManager class (continued)

addReaderGroupMember (CollabSpaceBean csbean)	Any of the following: • addCWMembers(CollabWorkspaceInfo csbean) • addCWMembers(String collabWorkspaceId,Vector memberDNs,String role) • addCWMember(String collabWorkspaceId,String memberDN,String role)
createCollabSpace (CollabSpaceBean cspaceBean, java.lang.String creatorDN)	createCollabWorkspace(CollabWorkspaceInfo cspaceBean)
deleteAuthorGroupMember (CollabSpaceBean csbean)	Any of the following: • removeCWMembers(CollabWorkspaceInfo csbean) removeCWMembers(String cwId, Vector memberDNs) • removeCWMember(String cwId, String memberDN)
deleteCollabSpace (java.lang.String collabSpaceID)	removeCollabWorkpace(String collabSpaceID)
deleteManagerGroupMember (CollabSpaceBean csbean)	Any of the following: • removeCWMembers(CollabWorkspaceInfo csbean) • removeCWMembers(String cwId, Vector memberDNs) • removeCWMember(String cwId, String memberDN)
deleteReaderGroupMember (CollabSpaceBean csbean)	Any of the following: • removeCWMembers(CollabWorkspaceInfo csbean) • removeCWMembers(String cwId, Vector memberDNs) • removeCWMember(String cwId, String memberDN)
deleteCollabSpace (java.lang.String collabSpaceID)	getCollabWorkspaceDetails(String collabSpaceID)
isUserDNAuthorGroupMember (java.lang.String userDN, java.lang.String collabSpaceID)	getCWMemberRole(String memberDN, String cwId)

Table 5. API mapping between CollabManagerBean class and CollabManager class (continued)

getCWMemberRole (String memberDN, String cwId)	getCWMemberRole(String memberDN, String cwId)
isUserDNReaderGroupMember (java.lang.String userDN, java.lang.String collabSpaceID)	getCWMemberRole(String memberDN, String cwId)
listAllCollabSpaces()	listCollabWorkspaces()
listAuthorGroupMembers (java.lang.String collabSpaceID)	listCWMembers(String collabSpaceID, String role)
listCollabSpaceForUserDN (java.lang.String userDN)	listCollabWorkspaces(String userDN)
listCollabSpaceTemplates()	listCWTemplates()
listCWTemplates()	listCWTemplates()
listReaderGroupMembers (java.lang.String collabSpaceID)	listCWMembers(String collabSpaceID, String role)
modifyAuthorGroupMembers (CollabSpaceBean csbean)	setCWMemberRole(String memberDN, String cwId, String role)
modifyCollabSpaceDescription (java.lang.String collabSpaceID, java.lang.String description)	changeCWDescription(String collabSpaceID,String description)
changeCWDescription (String collabSpaceID,String description)	changeCWDescription(String collabSpaceID,String description)
changeCWDescription (String collabSpaceID,String description)	setCWMemberRole(String memberDN, String cwId, String role)
TESTaddAuthorGroupMembers (java.lang.String csID)	No applicable; no equivalent
TESTaddAuthorGroupMembers (java.lang.String csID, java.lang.String userDN) TESTaddManagerGroupMembers (java.lang.String csID)	No applicable; no equivalent
TESTaddManagerGroupMembers (java.lang.String csID, java.lang.String userDN)	No applicable; no equivalent
TESTaddReaderGroupMembers (java.lang.String csID) TESTaddReaderGroup Members(java.lang.String csID, java.lang.String userDN)	No applicable; no equivalent

Table 5. API mapping between CollabManagerBean class and CollabManager class (continued)

TESTcreateCollabSpace (java.lang.String bProcType, java.lang.String bProcID)	No applicable; no equivalent
TESTdeleteAuthorGroupMembers (java.lang.String csID)	No applicable; no equivalent
TESTdeleteAuthorGroupMembers (java.lang.String csID, java.lang.String userDN)	No applicable; no equivalent
TESTdeleteManagerGroupMembers (java.lang.String csID)	No applicable; no equivalent
TESTdeleteManagerGroupMembers (java.lang.String csID, java.lang.String userDN)	No applicable; no equivalent
TESTdeleteReaderGroupMembers (java.lang.String csID)	No applicable; no equivalent
TESTdeleteReaderGroupMembers (java.lang.String csID, java.lang.String userDN)	No applicable; no equivalent
TESTmodifyAuthorGroupMembers()	No applicable; no equivalent
TESTmodifyManagerGroupMembers()	No applicable; no equivalent
TESTmodifyReaderGroupMembers()	No applicable; no equivalent

The following table lists other miscellaneous API mappings for collaborative workspaces:

Table 6. Other API mappings for collaborative workspaces

Method in CollabManagerBean	**Equivalent implementation**
getCollabAppServerIP()	WcsApp.configProperties.getValue (ECConstants.EC_COLLAB_QP_HOST)
getCollabAppServerPort()	WcsApp.configProperties.getValue (ECConstants.EC_COLLAB_QP_HTTP_PORT)

4. Since the CollabManager is not an enterprise bean, remove all the try and catch entries previously used to for enterprise bean coding to handle exceptions raised by enterprise beans. Remove the try and catch entries to avoid compling errors.
5. Compile your code and fix any compiling errors that are raised by your IDE.

Request for Quotation (RFQ)

Business A Request For Quotation (RFQ) is a process where a buying organization solicits offers from selling organizations to obtain a suitable price for a given product or set of products. The buyers in the buying organization may not find the products they wanted in the catalog of the selling organizations. In WebSphere Commerce 5.5, the RFQ request tools have been changed to allow the buyer to create an RFQ on a ″made to order″ item. To fully specify the attributes of the ″made to order″ item, personalization attributes are used just like any other RFQ item. For details on RFQ migration, including new or changed (command or database table) assets and how to migrate the RFQ request tool, refer to ″Request For Quotation (RFQ) migration″ section within the *WebSphere Commerce Migration Guide*.

In addition, for information on supported, deprecated, reserved, and withdrawn API, refer to the WebSphere Commerce 5.5 Development online help.

Database schema differences

For details on the database schema information for WebSphere Commerce and WebSphere Commerce development environment, refer to the WebSphere Commerce Development online help. Once you have launched the online help, select **WebSphere Commere Development information** > **Reference** > **Data** > **Database schema**. From here, select **Database tables** to view an alphabetical list of all database schema information. Select **Database changes in this release** to view the schema changes for Version 5.5. The online help also provides data model information.

Ensure that you visit the WebSphere Commerce Technical Library (http://www.ibm.com/software/commerce/library/), for the latest versions of the WebSphere Commerce documentation, including any updates to the database schema information in the online help.

API differences

For details on the API information for WebSphere Commerce and WebSphere Commerce development environment, refer to the WebSphere Commerce Development online help. Once you have launched the online help, select **WebSphere Commere Development information** > **Reference** > **API information**. The online help also provides API information on the following:

- Which API are supported for Version 5.5.
- Which API have been changed for Version 5.5. If you used or customized any of these API, ensure that you review the changed versions. With version 5.5 of WebSphere Commerce and WebSphere Commerce development environment, you will have migrated to enterprise bean level

1.1. For more information on this and enterprise bean Finder Helper implementation, refer to "Migrating enterprise beans from 1.0 to 1.1 specification" and "Differences between VisualAge for Java and WebSphere Studio Application Developer" on page 45.

- Which API have been reserved for IBM internal use and are for reference purposes only.
- Which API have been deprecated for Version 5.5. If you have used any of the deprecated API, refer to the online help for the replacement API to use with Version 5.5.
- Which API have been withdrawn from Version 5.5 and are no longer supported.
- Which API are no longer compatible with Version 5.5, but were supported for Version 5.1 and 5.4.

Cross reference information is also inlcude for data beans, enterprise beans, and database tables; and commands, tasks, and database tables.

Ensure that you visit the WebSphere Commerce Technical Library (http://www.ibm.com/software/commerce/library/), for the latest versions of the WebSphere Commerce documentation, including any updates to the API information the online help.

Migrating enterprise beans from 1.0 to 1.1 specification

Transitioning from 1.0 to 1.1 specification does not impact the access bean layer or your code; you can use the code you used previously, with your WebSphere Commerce development environment. Transitioning from 1.0 to 1.1 does impact your enterprise beans. The following table illustrates the enterprise bean changes that were made for the WebSphere Commerce development environment, and thus will impact your migration from 1.0 ot 1.1 specification.

Table 7. Enterprise bean changes and impact to the WebSphere Commerce development environment

Change made for the WebSphere Commerce development environment, version 5.5	How the change impacts your enterprise beans
Rename the enterprise bean ″impl″ to ″bean″ base class	No impact as this is a change that involves naming and packaging conventions.
Remove ″implements java.io.Serializable″ from BeanBase class	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration.

Table 7. Enterprise bean changes and impact to the WebSphere Commerce development environment (continued)

Remove serial version UID	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration.
Ensure ejbCreate() returns primary key object	You must make this change to comply with the enterprise bean 1.1 specification.
Remove RemoteException from ejbCreate(), ejbPostCreate(), ejbActivate(), ejbPassivate(), ejbStore(), ejbRemove(), and all remote methods	You must make this change to comply with the enterprise bean 1.1 specification.
Remove getEntityContext(), setEntityContext(), unsetEntityContext() from the bean or bean base class	No impact as this is an internal cleanup fix.
Move all ejbCreate() and ejbPostCreate() code to the bean base class	No impact as this is an internal cleanup fix.
Match ejbPostCreate() and ejbCreate() methods	You must make this change to comply with the enterprise bean 1.1 specification.
Add the WCSecurity role	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration. The enterprise bean security role is needed if security is turned on within WebSphere Studio Application Developer.
Specify the enterprise bean transaction setting as ″Required″	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration. This enterprise bean transaction setting is required. The value depends on the enterprise bean usage.

Table 7. Enterprise bean changes and impact to the WebSphere Commerce development environment (continued)

Specify the enterprise bean isolation level as ″Repeatable Read″ for each enterprise bean	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration. This enterprise bean isolation level is required. The value depends on the database type used.
Change the JNDI name	No impact as this is a change that involves naming and packaging conventions.
Delete xyzBeanFinderHelper interface	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration. The BeanFinderHelper interface is obsolete. All finders are defined in ibm-ejb-jar-ext.xmi.
Rename xyzBeanFinderObjectImpl to xyzBeanFinderObjectBase	Although this change is not directly related to the enterprise bean 1.1 specification, you will need to make this change for your enterprise beans to work properly after migration. The BeanFinderHelper interface is obsolete. All finders are defined in ibm-ejb-jar-ext.xmi.

Checking the unique index in the MSGTYPES table

The MSGTYPE_ID and NAME column are new for the MSGTYPES table. The MSGTYPE_ID column is a primary key and the NAME column is a unique index. If you have created your own message types, ensure that they do not conflict with the new ones. If they do, you can modify the MSGTYPE_ID value by adding 10,000 to the numbers or modify the NAME value. You will need to re-compile your commands that reference to the custom message types. Otherwise, you may encounter problems when migrating your data from WebSphere Commerce Suite 5.1 or from WebSphere Commerce 5.4

WebSphere Application Command Framework

For WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.4, the San Francisco Command Framework was used. WebSphere Commerce 5.5, however, has changed to use the WebSphere Application Command Framework. Thus, when migrating custom or extended code, it is important to note the following:

- All the customer commands that extend from WebSphere Commerce Suite 5.1 or WebSphere Commerce 5.4 controller commands or task command are required to be recompiled in order to pick up the new WebSphere Application Server, Version 5 runtime libraries.
- The callers of the controller commands and task commands are required to have the command context set.

J2EE Connector Architecture

For WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.4, the Common Connector Framework was used. WebSphere Commerce 5.5, however, has changed to use the J2EE Connector Architecture. If you used a connector with WebSphere Commerce Suite 5.1 or WebSphere Commerce 5.4, you will need to rewrite it. For details on J2EE framework and the J2EE Connector Architecture, refer to the follwing Web site from Sun Microsystem's: http://java.sun.com/j2ee/connector.

In addition, WebSphere Commerce Studio, Version 5.5 also provides an example connector called the Sample Adapter, which is a sample J2EE Connector Architecture adapter (connector).

Part 2. Migrating your WebSphere Commerce development environment

This part describes a typical migration flow to migrate your WebSphere Commerce development environment. The migration process involves a series of steps. Refer the following chapters within this part for instructions on completing each migration step.

Chapter 4. Migrating your WebSphere Commerce development environment

This chapter describes the detailed steps to upgrade your WebSphere Commerce development environment

Note: Before you upgrade your WebSphere Commerce development environment, ensure that you have reviewed the information outlined in Part 1, "Pre-migration," on page 1, including the hardware and software requirements outlined. Also, if you are migrating from WebSphere Commerce Version 5.1, ensure that you have a working VisualAge for Java 3.5.3 WebSphere Test Environment before you begin the migration. This ensures you have all required JSP files, commands, data beans, and enterprise beans before you begin migration.

Backing up WebSphere Commerce and WebSphere Commerce development environment assets

Before you upgrade any software or transition any custom code, ensure you back up both WebSphere Commerce and the WebSphere Commerce development environment assets, as follows:

- Back up the following WebSphere Commerce assets:
  - WebSphere Commerce directory tree
  - WebSphere Commerce database

  For the steps to back up the WebSphere Commerce directory tree and database, see the section ″Backing up Commerce Suite 5.1″ or ″Backing up WebSphere Commerce 5.4″ in the *WebSphere Commerce Migration Guide* for the version of WebSphere Commerce that you are migrating from.
- Back up the WebSphere Commerce development environment assets.

Upgrading your DB2 product level

If you use DB2 as your database management system, after you have backed up your WebSphere Commerce and WebSphere Commerce development environment assets, you must upgrade to and configure DB2 Version 8.1 before upgrading your WebSphere Commerce development environment. To install and configure DB2 Version 8.1 as the WebSphere Commerce development environment database, follow these high-level steps:

1. Record the configuration settings for your database before DB2 migration.
2. Change the diagnostic error level.

3. Take the DB2 server offline for DB2 migration.
4. Verify that databases are ready for DB2 migration.
5. Back up your databases.
6. *Optional*: If you will be using replication, you must archive all of the DB2 log files.
7. Install DB2 Enterprise Server Edition Version 8.1.
8. Migrate your databases.
9. *Optional*: Migrate DB2 Explain tables.

For details on the above steps, see Part 2. "Migrating your DB2 server" in the *Quick Beginnings for DB2 Servers* document. This document is available from the DB2 Technical Spport Library (http://www.ibm.com/software/data/db2/library/); you will need to navgate to the DB2 Technical Support Version 8 Information Center.

Upgrading your Oracle product level

If you use Oracle as your database management system, after you have backed up your WebSphere Commerce and WebSphere Commerce Studio assets, you must upgrade to and configure Oracle9i Database Release 2 (that is, version 9.2.0.1.0) before upgrading to WebSphere Commerce Studio, Version 5.5. To install and configure Oracle9i as theWebSphere Commerce Studio database, do the following:

1. Install the following Oracle9i Database Release 2 (9.2.0.1.0) components according to the instructions found in the Oracle9i documentation:
   - Oracle9i Database
   - Oracle Net Services
   - Oracle Net Protocol Support
   - SQL*Plus
   - Oracle JDBC Thin Interfaces
   - Oracle JDBC/OCI Interfaces

Upgrading your WebSphere Commerce development environment

Note: It is important that you do not uninstall VisualAge for Java or an earlier version of WebSphere Studio Application Developer which you used for your previous version of your WebSphere Commerce development environment, as you will need to export your assets and projects from these applications into WebSphere Studio Application Developer, Version 5. However, to avoid conflict, ensure that you upgrade your WebSphere Commerce development environment to a different directory than that of VisualAge for Java or an earlier version of WebSphere Studio Application Developer. Note also that your VisualAge for Java projects will not work on the WebSphere Studio Application Developer workspace.

To upgrade WebSphere Commerce development environment, Version 5.5, do the following (for detailed installation instructions, refer to *WebSphere Commerce Studio Installation Guide*):

1. Insert the WebSphere Commerce development environment CD in your CD-ROM drive. If you have autoplay enabled, the installation process begins. If not, run setup.exe.
2. Remove the \WebSphere\Commerce\bin directory path from the system environment path. To remove this directory, do the following:
   a. Access your **System Properties** window.
   b. Select **Environment Variables**. Under **System variables**, select **Path**.
   c. Edit this path and remove the old WebSphere Commerce development environment path (for example, WebSphere\Commerce\bin).
   d. Click **OK**.
3. Follow the prompts in the installation wizard.
4. Once the WebSphere Commerce development environment installation has completed, and you have custom code created in VisualAge for Java, the custom code needs to be migrated to WebSphere Studio Application Developer.

   Note: This step is only required if you are migrating back-end assets such as commands or data beans.

   To ease this process, start VisualAge for Java and export the following projects and groups now to JAR files as outlined in "Exporting task commands, controller commands, and data beans to a JAR file" and "Exporting enterprise beans to a JAR file" on page 38. (Details on transitioning custom code and logic are described in "Differences between VisualAge for Java and WebSphere Studio Application Developer" on page 45.)

Exporting task commands, controller commands, and data beans to a JAR file

Start VisualAge for Java and export the VisualAge for Java project containing custom code for new or extended task commands, controller commands, or data beans. In this case, you must export the project or projects containing this code to a JAR file as follows:

1. In the VisualAge for Java Workbench window, select and right-click the project name or names, then click **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.

Exporting enterprise beans to a JAR file

Start VisualAge for Java and export any custom enterprise beans as follows to a JAR file (you must export all the enterprise bean groups containing custom enterprise beans to a single JAR file).

If you have minimal custom enterprise beans

If you have a minimal number of custom enterprise beans (that is, approximately ten or less), it is recommended that you re-create the enterprise beans using WebSphere Studio Application Developer. For details on how to do this, refer to the *WebSphere Studio Application Developer, Version 5 Migration Guide*

If you have many custom enterprise beans

Migrating from WebSphere Commerce Studio, Version 5.1: If you are migrating from WebSphere Commerce Studio, Version 5.1 and you have a large number of custom enterprise beans, here are the high level steps to follow:

1. Export your enterprise beans from VisualAge for Java 3.5.3 as a repository.
2. Contact WebSphere Commerce Support to obtain a copy of VisualAge for Java 4.0. It is recommended that you install VisualAge for Java 4.0 on a single machine as the transition machine for migrating enterprise beans.
3. Prepare your VisualAge for Java 4.0 environment as follows:
   a. Ensure that the WebSphere Commerce Studio, Version 5.1 repository is within the VisualAge for Java 4.0 environment.
   b. Ensure that you have IBM EJB Development Environment 4 installed.
   c. In VisualAge for Java 4.0, add the **Export Tool for Enterprise Java Beans 1.1** feature to the workspace.
   d. Import the repository that you exported from VisualAge for Java 3.5.3 to VisualAge for Java 4.0.
4. In VisualAge for Java 4.0, export each of your enterprise bean groups containing custom enterprise beans 1.1 to a corresponding JAR file:

   If during the export process, the **Target database type** selection box is disabled, this is an indication that you have lost your enterprise schema mapping.

   a. On the EJB page of the Workbench, right-click an enterprise bean group and click **Export** > **EJB 1.1 JAR**.

    b. Fill in the fields as necessary. Ensure that the **.java** check box is selected.
    c. Ensure that you select your database.
    d. Click **Finish**.

    Note: Multiple enterprise bean groups cannot be exported to a single JAR file. If you have multiple EJB groups, do the following in VisualAge for Java 4.0:

    a. Open the **Projects** tab
    b. Export your custom EJB projects as repositories one-at-a-time.
    c. Create a new project.
    d. Go to the EJB tab and create an EJB group in this project.
    e. Import your custom EJB project repositories that you exported in step 4b into this EJB group. If your project has been versioned, ensure that you select the option to create new or scratch editions.

5. Import the JAR file into WebSphere Studio Application Developer, Version 5.

Migrating from WebSphere Commerce Studio, Version 5.4: If you are migrating from WebSphere Commerce Studio, Version 5.4 and you have a large number of custom enterprise beans, you can do this by using the Export Tool for Enterprise Java Beans 1.1. Export all the enterprise bean groups containing custom enterprise beans 1.1 to a single JAR file as follows:

If during the export process, the **Target database type** selection box is disabled, this is an indication that you have lost your enterprise schema mapping.

1. In VisualAge for Java, add the **Export Tool for Enterprise Java Beans 1.1** feature to the workspace if you have not already added it.
2. On the EJB page of the Workbench, right-click 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**.
6. Import the JAR file into WebSphere Studio Application Developer, Version 5.

Applying WebSphere Commerce fix pack 2

After you upgrade the WebSphere Commerce development environment, you need to install the WebSphere Commerce fix pack 2, so that you are at the 5.5.0.2 level. according to the ″Applying the fix pack to WebSphere Commerce Studio″ section of the guide provided with the fix pack.

This fix pack includes fixes and enhancements for the migration process. This fix pack is available from the WebSphere Commerce Support site. Click on the link for the edition you are using and see the section Software downloads. Download this fix pack, and follow the instructions that are provided to install it. You should not proceed with the migration steps in this guide (such as migrating your instance or database) without installing this fix pack.

Migrating the WebSphere Commerce development environment instance

During the WebSphere Commerce instance migration process, the following tasks are completed:

- Migrates the instance.xml file.
- Copies the store front or Web assets from VisualAge for Java into a WebSphere Studio Application Developer Web project. This includes any JSP files, HTML pages, or graphics you used with your store front.
- Extracts the JAR file containing the non-enterprise bean custom code into the WebSphere Studio Application Developer Java project called WebSphereCommerceServerExtensionsLogic.
- Migrates the JSP files that were copied when you run the database migration script, to conform to the JSP 1.2 specification.

Note: Instance migration involves automatically copying the store front or Web assets (that is, JSP files, HTML pages, or graphics) from VisualAge for Java to WebSphere Studio Application Developer. However, you should also ensure that you have accounted for all the assets created for your store front within your previous version your WebSphere Commerce development environment.

Once you have applied WebSphere Commerce fix pack 2, you can migrate the WebSphere Commerce development environment instance, as follows:

1. If you are migrating store front assets, start WebSphere Studio Application Developer and create a Web project for your store assets and note the name of the project. You will need to have this project created before running the wstudioim.bat instance migration script as you will need to specify the Web project name as a parameter value when running this script.

Note: If you only are only migrating back-end assets, then you can skip this step.

2. Ensure that all the values are properly set for your migration path within the product.xml file found in the *WCDE_installdir*\xml directory. The values for product.xml get filled in by the WebSphere Commerce 5.5 installation program, however, it is recommended that you check the values to ensure they are correct for your environment. Specifically, ensure that the <migrationFrom> element and all the child elements of this element contain values. If they do not, you must manually insert the values. The values can be obtained from the product.xml file used with your previous version of WebSphere Commerce; refer to the <CommerceServer> tag.

   The <migrationFrom> element should be in the following format:

```
<migrationFrom>
  <edition>
    <name>name</name>
  </edition>
  <version>5</version>
  <release>rel</release>
  <modification>mod</modification>
  <fixpak>fixpak</fixpak>
  <path>path</path>
  <altpath>long_path</altpath>
</migrationFrom>
```

   where:

   name The edition of WebSphere Commerce from which you are migrating. . The name can be any of the following:
   - **Start** — Denotes migration from WebSphere Commerce Start Edition
   - **Pro** — Denotes migration from WebSphere Commerce Professional Edition
   - **Business** — Denotes migration from WebSphere Commerce Buesiness Edition

   rel The release of the WebSphere Commerce development environment from which you are migrating. The release will be **1** for migration from WebSphere Commerce Suite 5.1.

   mod The modification level of the WebSphere Commerce development environment from which you are migrating. The modification level is always a value of **0**.

   fixpak The fix pack level of WebSphere Commerce from which you are migrating. For example, if you are migrating from WebSphere Commerce Suite 5.1.0.**1**, the fixpak level should be **1** in the product.xml file.

path The installation path for the version of WebSphere Commerce from which you are migrating. This is typically the same as the *long_path* value. For example, if you are migrating from WebSphere Commerce Suite 5.1, the path is *WC51_installdir*, such as *c:\WebSphere\WCS*.

long_path
The full or long installation path for the version of WebSphere Commerce from which you are migrating. This is typically the same as the *path* value. For example, if you are migrating from WebSphere Commerce Suite 5.1, the path is *WC51_installdir*, such as *c:\WebSphere\WCS*.

For your reference, the following is an example of what your product.xml file needs to contain if you are migrating from WebSphere Commerce 5.1.0.1 Start Edition:

```
<migrationFrom>
  <edition>
    <name>Start</name>
  </edition>
  <version>5</version>
  <release>1</release>
  <modification>0</modification>
  <fixpak>1</fixpak>
  <path>c:\WebSphere\WebSphere Commerce</path>
  <altpath>c:\WebSphere\WebSphere Commerce</altpath>
</migrationFrom>
```

3. Edit the *WC_installdir*\bin\wstudioimenv.bat file and ensure the values for the following settings are correct:
   - SET WCIM_MIGRATE_FROM should be SET WCIM_MIGRATE_FROM=51 if you are migrating from WebSphere Commerce Suite 5.1 and SET WCIM_MIGRATE_FROM=54 if you are migrating from WebSphere Commerce 5.4SET WCIM_MIGRATE_FROM=51SET WCIM_MIGRATE_FROM=54
   - SET WC_PATH is set to the *WC_installdir*; that is, the directory of where you installed WebSphere Commerce 5.5. For example, C:\WebSphere\CommerceStudio\Commerce
   - SET ANT_PATH is set to *WSAD_installdir*\runtimes\base_v5\lib
   - SET WORK_DIR is set to a temporary working directory for *WCDE_installdir*; for example, *WCDE_installdir*\commerce\temp
   - SET LOG_FILE is set to the name of the file to log information about the instance migration.
   - SET INSTANCE is the name of the WebSphere Commerce 5.5 instance.
4. From a command prompt, change to the *WCDE_installdir*\bin directory and run the wstudioim script by typing the following as one line:

```
wstudioim.bat WSAD_workspacedir VAJ_installdir
  custom_code_JAR_file store_properties_dir
```

Note: If you are only migrating store front assets, for the custom_code_JAR_file parameter, specify a blank value (that is, use two double quotation marks to indicate the blank value). For example, in this case, you can execute the instance migration script as follows:

```
wstudioim.bat WSAD_workspacedir VAJ_installdir ""
store_properties_dir
```

where:

WSAD_workspacedir
: The path for your current WebSphere Studio Application Developer workspace, specified when you upgraded your WebSphere Commerce development environment. For example, c:\WebSphere\workspace_db2.

VAJ_installdir
: The installation path for VisualAge for Java. The default installation path is \Visual Age for Java. For example, d:\Visual Age for Java.

custom_code_JAR_file
: The JAR file from VisualAge for Java, which contains custom code such as customized enterprise beans, commands, or data beans. For example, d:\myCustomCode.jar.

store_properties_dir
: The path for store properties files of your previous version of WebSphere Commerce. For example, if you are migrating from WebSphere Commerce Suite 5.1, an example path is c:\WebSphere\WCS\stores\properties.If you are migration from WebSphere Commerce 5.4, an example path is c:\WebSphere\CommerceServerDev\wc.ear\wcstores.war \WEB-INF\classes.

5. Edit the migrated instance XML file as follows:
   a. Search for WorkspacePath=″″
   b. Add the WebSphere Commerce development environment workspace path between the quotation marks (for example, WorkspacePath=″D:\WEBSPH~1\WORKSP~1″).
   c. Add the new WebSphere Commerce development environment workspace path between the quotation marks (for example, WorkspacePath=″D:\WEBSPH~1\WORKSP~2″).
6. If you are migrating from WebSphere Commerce Studio, Version 5.1 to WebSphere Commerce Studio, Version 5.5, in order to log onto the WebSphere Commerce Accelerator, Administration Console, or Organization Administration Console, using the logon ID and password you used previously, after you have migrated your WebSphere Commerce

instance, you must modify the merchant key by running the MKChangeDBUpdate utility. For details on how to run this utility, refer to the "Migrating your security configuration" section of the *WebSphere Commerce Migration Guide*.

7. Import your custom enterprise beans to the WebSphereCommerceServerExtensionsData enterprise bean project within WebSphere Studio Application Developer. Start WebSphere Studio Application Developer and import the enterprise bean JAR file or files exported from VisualAge for Java in step "Exporting enterprise beans to a JAR file" on page 38 as follows:
   a. In WebSphere Studio Application Developer, from the J2EE perspective, select **File** > **Import** > **EJB JAR file**. Click **Next**.
   b. Specify the JAR file containing your enterprise beans.
   c. Specify the enterprise bean project where the enterprise beans will be imported by selecting the existing project called **WebsphereCommerceServerExtensionsData**.
   d. Click **Next** and select all the enterprise bean files from your JAR file.
   e. Click **Next** and select all the dependent projects. If you are unsure as to which projects you have a dependency on, select all of them.
   f. Click **Finish**. 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 52 to troubleshoot them. After you have imported your enterprise beans, follow the instructions in "Locating enterprise bean information" on page 53 to determine where to find your enterprise bean and method properties, your schemas, maps, and so on.

Migrating the WebSphere Commerce development environment database schema

To migrate the WebSphere Commerce development environment database schema, you run a database schema migration script: either migratedb51.bat (if you are migrating from WebSphere Commerce Studio, Version 5.1) or migratedb54.bat (if you are migrating from WebSphere Commerce Studio, Version 5.4) as follows:

__ 1. Before running the database schema migration script, ensure that the JAVA_HOME environment variable in the setenv.bat file points to an existing version of IBM Developer Kit, Java Technology Edition on your machine. The setenv.bat file is normally located in the *WCDE_installdir*\Commerce\bin directory.

For example, if IBM Developer Kit, Java Technology Edition is installed in the \eclipse\jre directory on your WebSphere Commerce development environment machine, set `JAVA_HOME` to `JAVA_HOME=`*`WCDE_installdir`*`\java\jre`. By default, JAVA_HOME points to the WebSphere Commerce development environment installation directory (for example, `JAVA_HOME=C:\WebSphere\Studio5\runtimes\base_v5\java\jre`).

__ 2. Before running the database schema migration script, ensure that you drop all customized foreign key references. This will prevent various database errors that you may encounter during migration. After the database schema migration, you should rebuild these foreign key references manually.

__ 3. To run the database migration script, see the section ″Migrating your database″ in the *WebSphere Commerce Migration Guide*.

Notes:

a. In the WebSphere Commerce development environment, the database schema migration scripts are available in the *WCDE_installdir*\commerce\bin directory.

b. You can run the premigratedb script, as described in *WebSphere Commerce Migration Guide*, you can do so from a command prompt; whereas the migratedb script should be run from a DB2 Command Line Processor window.

Transitioning your customized or extended code

Once you have migrated the WebSphere Commerce instance, your next step is to transition your extended or modified business logic and code contained within your VisualAge for Java workspace to the WebSphere Studio Application Developer level. To help with the complex task of transitioning custom or extended code, two tutorials are provided within this guide. Refer to Part 4, “Tutorials for transitioning your code,” on page 67 to learn more about migrating customized or extended code.

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 highlights the main differences between VisualAge for Java and WebSphere Studio Application Developer:

- The Enterprise Java Beans (EJB) specification level has changed from 1.0 to 1.1. That is, the 1.1 specification for the version 5.5 WebSphere Commerce development environment using WebSphere Studio Application Developer, Version 5.
- For Web applications, the JSP level changed from 1.1 to 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.)
- The Visual Composition Editor has been replaced by the Visual Editor for Java.
- 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.
- Convertors in VisualAge for Java EJB AccessBeans (not Data AccessBeans) are not available in WebSphere Studio Application Developer EJB AccessBeans. Use enterprise bean convertors and composers in the underlying enterprise bean instead. For more information about enterprise bean convertors and composers, refer to the online help documentation.

VisualAge for Java 3.5.3 enterprise bean 1.0 JARs versus VisualAge for Java 4.0 enterprise beans 1.1 JARs

For more details about migrating your enterprise beans from VisualAge for Java 3.5.3, refer to the *WebSphere Studio Application Developer, Version 5 Migration Guide*.

If you are migrating from WebSphere Commerce Studio, Version 5.1, you used VisualAge for Java 3.5.3 enterprise bean 1.0 JARs. Simply migrating the custom enterprise beans used with WebSphere Commerce Studio, Version 5.1 and VisualAge for Java 3.5.3 to WebSphere Commerce Studio, Version 5.5 and WebSphere Studio Application Developer will cause you to lose some information. The solution is to successfully migrate your enterprise beans is to use of the VisualAge for Java 4.0 EJB Export Tool.

In VisualAge for Java 3.5.3 you can generate an undeployed enterprise bean 1.0 JAR, or generate a deployed enterprise bean 1.0 JAR (with deployment code, stubs, tie code, and so on). You can import the enterprise bean 1.0 JAR into the WebSphere Application Server 4.0 Application Assembly Tool (AAT) and use that to generate deployment and control descriptors and to run the EJB Deploy Tool to generate deploy code, stubs, and so on into an enterprise bean 1.1 format JAR (enterprise applications archives). The net result of the previous steps is that you can have a deployed JAR that can be run in WebSphere Application Server Version 4.0, and hence can also be run in WebSphere Studio Application Developer (since its WebSphere Test

Environment is a WebSphere Application Server 4.0 server). However, run-time execution does not imply ongoing development capability.

You can include the previously generated Java code into the corresponding enterprise bean JARs and import that into WebSphere Studio Application Developer, and WebSphere Studio Application Developer would parse and understand the enterprise bean session support, security information, finder information, and assembly descriptor (const methods, and so on). However, extensions (class inheritance and ForeignKey associations) and the map and schema information are all lost. They can only be retained if you use the VisualAge for Java 4.0 EJB Export Tool, to create an enterprise bean 1.1 JAR containing the map and schema XML information and extension XML information. As well, the names of generated stub classes are different in 1.0 than 1.1 enterprise beans. The net result is that neither an enterprise bean 1.0 JAR nor an EJB Deploy Tool enterprise bean 1.1 JAR retains the basic design meta-data to allow them to be used for ongoing development in WebSphere Studio Application Developer.

The VisualAge for Java 4.0 EJB Export Tool produces an enterprise bean 1.1 JAR that contains all the enterprise bean design meta-data, including the map and schema information and the extensions (class inheritance and ForeignKey associations), so that you can immediately continue to develop within WebSphere Studio Application Developer and regenerate deployment code anytime that is required. This is the only supported method of migrating VisualAge for Java enterprise beans into WebSphere Studio Application Developer; and the use of the VisualAge for Java 4.0 EJB Export Tool is the only method that will work for ongoing enterprise bean development within WebSphere Studio Application Developer.

Transitioning extended or modified WebSphere Commerce business logic and code

When you migrate your WebSphere Commerce development environment, a number of steps related to your VisualAge for Java environment are performed. These steps are as followings:

- 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 your enterprise beans into WebSphere Studio Application Developer.
- Migrating your customized WebSphere Commerce Server enterprise beans into WebSphere Studio Application Developer.
- Setting up your test environment and testing your migrated application.

Notes:

1. For WebSphere Commerce Suite 5.1 and WebSphere Commerce 5.4, the Common Connector Framework was used. WebSphere Commerce 5.5, however, has changed to use the WebSphere Application Command Framework. Thus, when migrating custom or extended code, it is important to note the following:
   - All the customer commands that extend from WebSphere Commerce Suite 5.1 or WebSphere Commerce 5.4 controller commands or task command are required to be recompiled in order to pick up the new WebSphere Application Server, Version 5 runtime libraries.
   - The callers of the controller commands and task commands are required to have the command context set.
2. For information on supported, deprecated, reserved, and withdrawn API, refer to the WebSphere Commerce 5.5 Development online help.

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

Your 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 command or a task command, into the Web project source directory (the proper hierarchy based on their package statements will automatically be created by WebSphere Studio Application Developer).
  - Import all resource files, such as JSP pages, into the Web project Web content directory.
- If your code is straight Java, you should import the code into a Java project.
- If your code is enterprise beans, you should import the code into an enterprise bean project.

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:

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.
4. Select the files you want to import and the project or folder you want to contain your files.

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. If you are migrating from WebSphere Commerce Studio, Version 5.1, you used VisualAge for Java 3.5.3. If you have a minimal number of customer enterprise beans, it is recommended that you re-create the enterprise beans using WebSphere Studio Application Developer, Version 5. For details on how to do this, refer to the *WebSphere Studio Application Developer, Version 5 Migration Guide*.

   If you have a large number of custom enterprise beans, you first need to export your enterprise beans from VisualAge for Java 3.5.3 to VisualAge for Java 4.0 (using the Export Tool for Enterprise Java Beans 1.1 as mentioned in the bullet below), then export to enterprise bean 1.1 JAR within VisualAge for Java 4.0, and finally import into WebSphere Studio Application Developer, Version 5. If you are a WebSphere Commerce Studio, Version 5.1 customer migrating to WebSphere Commerce Studio, Version 5.5, contact WebSphere Commerce Support to obtain a copy of VisualAge for Java 4.0. It is recommended that you install VisualAge for Java 4.0 on a single machine as the transition machine for migrating enterprise beans.
2. If you are migrating from WebSphere Commerce Studio, Version 5.4, then you can do this by using the Export Tool for Enterprise Java Beans 1.1. If you are a WebSphere Commerce Studio, Version 5.1 customer migrating to WebSphere Commerce Studio, Version 5.5 and have contacted WebSphere Commerce Support to obtain a copy of VisualAge for Java 4.0, you can also use the Export Tool. In both cases, you must export all the enterprise bean groups containing custom enterprise beans to a single JAR file as follows:
   a. In VisualAge for Java, add the **Export Tool for Enterprise Java Beans 1.1** feature to the workspace if you have not already added it.
   b. On the EJB page of the Workbench, right-click on your enterprise bean group or groups and click **Export** > **EJB 1.1 JAR**.

c. Fill in the fields as necessary. Ensure that the **.java** check box is selected.
d. Ensure that you select your database.
e. Click **Finish**.

Importing your enterprise beans into WebSphere Studio Application Developer:

1. In WebSphere Studio Application Developer, create a new EJB project and a new enterprise applications archives project. You will automatically be switched to the J2EE perspective.
2. Select **File** > **Import** > **EJB JAR file**. Click **Next**.
3. Select your JAR file, your EJB project, and your EAR project.
4. Click **Next** and select all the EJB files from your JAR file.
5. Click **Next** and select all the dependent projects. If you are unsure as to which projects you have a dependency on, select all of them.
6. Click **Finish**. 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 52 to troubleshoot them. After you have imported your enterprise beans, follow the instructions in "Locating enterprise bean information" on page 53 to determine where to find your enterprise bean and method properties, your schemas, maps, and so on.

Editing methods to avoid java.rmi.RemoteException errors:

1. In WebSphere Studio Application Developer, go to the J2EE perspective.
2. From the J2EE Navigator view, open WebSphereCommerceServerExtensionData\ejbModule\com.ibm.commerce.sample.objects\enterprise_bean

   where *enterprise_bean* is the name of the enterprise bean you imported into WebSphere Studio Application Developer; for example, MyEnterpriseBean.java.
3. Modify the following methods so that they do not throw java.rmi.RemoteException errors:
   ```
   ejbLoad()
   ejbStore()
   unsetEntityContext()
   ```
   For example, change `public void ejbLoad() throws java.rmi.RemoteException` to `public void ejbLoad()`
4. Save your changes.

Changing the access isolation level:

1. In WebSphere Studio Application Developer, go to the J2EE perspective.

2. From the J2EE Navigator view, open ejb-jar.xml under \WebSphereCommerceServerExtensionData\ejbModule\META-INF
3. Click **Access**.
4. Select the node **Committed**, which is below **Isolation Level**, and click**Remove**.
5. Click **Add**, select **Repeatable read**, and click **Next**.
6. Select the name of the enterprise bean and click **Next**.
7. Expand name of the enterprise bean and select the check box for all home methods (that is, the first icon under name of the enterprise bean).
8. Click **Finish**.
9. Verify that the node **Repeat** now appears under **Isolation Level**.
10. Save your changes.

Generating deploy code:

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**. Select to generate code for all your enterprise beans.

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 8. Locating enterprise bean information after migrating to WebSphere Studio Application Developer

Item	**Location**
Enterprise beans (fields, classes)	Expand the EJB Modules folder in the J2EE Hierarchy view.
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 Deployment Descriptor Editor**. In the EJB Deployment Descriptor Editor, select the **Beans** tab.

Table 8. Locating enterprise bean information after migrating to WebSphere Studio Application Developer (continued)

Item	Location
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 EJB 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 Deployment Descriptor**. In the EJB Deployment Descriptor , 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 your customized WebSphere Commerce Server enterprise beans into WebSphere Studio Application Developer

After migrating your enterprise beans into WebSphere Studio Application Developer, the next step in transitioning your code to WebSphere Studio Application Developer is to migrate your customized WebSphere Commerce

Server enterprise beans into WebSphere Studio Application Developer. This task involves two subtasks: migrating code and meta-data changes into existing WebSphere Commerce Server projects in WebSphere Studio Application Developer, and generating deploy code and the access bean.

Migrating code and meta-data changes into existing WebSphere Commerce Server projects in WebSphere Studio Application Developer: To migrate code and meta-data 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 above steps for each WebSphere Commerce Server enterprise bean project you want to change.

Note: You may encounter warnings or errors that during the code merge process. This is normal and any errors are fixed when you generate the deploy code and access bean, as described in "Generating deploy code" on page 81.

Generating deploy code

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 newly-created EJB JAR file and select **Generate** > **Deploy and RMIC Code**.
3. Select **All EJBs** and click **Finish**.

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 52

- “Locating enterprise bean information” on page 53
- “Migrating EJB access beans” on page 54
- “Migrating customer finder helpers” on page 54

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. In WebSphere Studio Application Developer, ensure that you close the local Web server or WebSphere Application Server server. Otherwise, the WebSphere Commerce will not start successfully.
2. Check the data source property page and change the default password to your password for the data source setting you used. By default, the **Default password** field is blank.
3. Provide the data source setting and ensure that you set the correct URL for the data source. If you do not ensure that the URL is correct, you will encounter errors when starting the Commerce Server and you will not be able to create a data source successfully. Depending on whether you use DB2 or Oracle as your database:
   - If you use DB2, the default JDBC data source URL is `jdbc:db2:`*`database_name`*; for example, `jdbc:db2:mall`. Ensure that the data source URL is such.
   - If you use Oracle, the default JDBC data source URL is `jdbc:oracle:thin:@`*`host_name`*`::null`. You must modify it to be `jdbc:oracle:thin:@`*`host_name`*`:`*`ORACLE_port`*`:`*`ORACLE_SID`*. For example, `jdbc:oracle:thin:@wcs212:1521:orcl`.
4. Update the the WebSphere Commerce Server data source with the correct database user password as follows:
   a. Start WebSphere Studio Application Developer by selecting **Start** > **Programs** > **Start WebSphere Studio** > **Application Developer 5.0**.
   b. In the J2EE Hierarchy view of the J2EE Perspective, expand **Servers** and double-click **WebSphereCommerceServer**. The WebSphereCommerceServer view displays.
   c. In the WebSphereCommerceServer view, select the **Data sources** page.
   d. On the Data sources page, expand **Server Settings**.
   e. In the **JDBC provider list** table, select **DB2 JDBC Driver** if you are using DB2 as the WebSphere Commerce development environment database, or select **Oracle JDBC Thin Driver** if you are using Oracle.

After selecting a JDBC provider, the tables below the JDBC provider list table updates.

f. In the **Data source defined in the JDBC provider selected above** table, select **jdbc/WebSphere Commerce DB2 DataSource** *instance_name* if you are using DB2 as the WebSphere Commerce development environment database, or **select jdbc/WebSphere Commerce Oracle DataSource** *instance_name* if you are using Oracle.

   where *instance_name* is the name of the WebSphere Commerce development environment instance. The default instance name is Demo_Dev.

g. Click **Edit**. The Modify Data Source wizard opens.

h. Ensure that the data source JDNI name is as follows, depending on whether you use DB2 or Oracle as your database and depending on whether you are migrating from WebSphere Commerce Suite 5.1 or from WebSphere Commerce 5.4. If the data source JDNI name value is not as such, the WebSphere Commerce Server will not start. In this case, you will need to modify the data source JDNI name to match the value within the instance.xml file.

   - If you are migrating from WebSphere Commerce Suite 5.1 and using DB2:
     ```
     jdbc/WebSphere Commerce DB2 Suite DataSource instance_name
     ```
     For example, if your WebSphere Commerce instance name is called **demo**, the JDNI name should be jdbc/WebSphere Commerce DB2 Suite DataSource demo.
   - If you are migrating from WebSphere Commerce Suite 5.1 and using Oracle:
     ```
     jdbc/WebSphere Commerce Oracle Suite DataSource instance_name
     ```
     For example, if your WebSphere Commerce instance name is called **demo**, the JDNI name should be jdbc/WebSphere Commerce Oracle Suite DataSource demo.
   - If you are migrating from WebSphere Commerce 5.4 and using DB2:
     ```
     jdbc/WebSphere Commerce DB2 DataSource instance_name
     ```
     For example, if your WebSphere Commerce instance name is called **demo**, the JDNI name should be jdbc/WebSphere Commerce DB2 DataSource demo.
   - If you are migrating from WebSphere Commerce 5.4 and using Oracle:
     ```
     jdbc/WebSphere Commerce Oracle DataSource instance_name
     ```

For example, if your WebSphere Commerce instance name is called **demo**, the JDNI name should be jdbc/WebSphere Commerce Oracle DataSource demo.

i. In the Modify Data Source wizard, replace any value in the **Default user password** field with the password for the ID shown in the **Default user ID** field.
j. Click **Finish** to update the information and close the Modify Data Source wizard.
k. Save the updates to the configuration by selecting **File** > **Save WebSphereCommerceServer**.

5. Go to the Servers view. If you cannot find it in the perspective, on the main menu select **Perspective** > **Show View** > **Servers**.
6. Right-click the server instance and select **Publish**.
7. When the publishing is complete, close the pop-up menu by clicking **OK**.
8. Right-click the server instance and select **Start**. The Console view displays.
9. Follow the startup process for the instance.

You are now ready to test your migrated code by publishing a store. Follow these steps to access and setup the store using the WebSphere Commerce Administration Console:

1. Open a browser window and type the following URL: https://*host_name*/webapp/wcs/admin/servlet/ToolsLogon?XMLFile=adminconsole.AdminConsoleLogonhttps
2. On the Administration Console Logon page, type `wcsadmin` as the user name, provide your password, and click **Logon**. (By default, the user name and password are both set to `wcsadmin` during installation, but you must change the password the first time you use the WebSphere Commerce Administration Console.)
3. After you log into the WebSphere Commerce Administration Console, select **Site**.
4. From the **Store** within the WebSphere Commerce Administration Console main page, select **Publish**.
5. Complete the information within this wizard, clicking **Next** to proceed through the pages.
6. On the Options page, click **Finish** to publish your store.

Note that in comparison to the WebSphere Test Environment in VisualAge for Java, you are able to perform HTTPS requests due to the set up you have just completed.

Migrating the payment instance and database

Your next step is to migrate the WebSphere Commerce payment instance and database to be used with WebSphere Commerce Payments (formerly Payment Manager). If you use an external payment system, then you do not have to complete this section, but you must ensure that your payment system works properly with Version 5.5.

For WebSphere Commerce 5.4, the payment database was required to be installed on a different machine than the one containing WebSphere Commerce. For WebSphere Commerce 5.5, you now have the option of installing the payment database locally on the same machine as WebSphere Commerce Studio, Version 5.5. Thus, for version 5.5, two scenarios for payments are possible:

- If you keep payments remote, refer to the *WebSphere Commerce Migration Guide* for details on how to migrate a Commerce Payments instance and database using the WebSphere Commerce Instance Migrator tool.
- If you want to install the payment database on the same machine that contains WebSphere Commerce Studio, Version 5.5, you must migrate your payment database and ensure that the payment instances created in WebSphere Studio Application Developer, Version 5 use the migrated payment database.

Part 3. Post-migration

This part describes step you should follow after you have migrated your WebSphere Commerce development environment.

Chapter 5. Post-migration actions

After you have completed the migration steps outlined in Part 2, “Migrating your WebSphere Commerce development environment,” on page 33, ensure that you have completed the following:

- The JavaServer Page templates in your store comply to the JavaServer Page 1.2 specification, created by Sun Microsystems. The JSP templates used in stores running in WebSphere Commerce Suite 5.1 were required to support the JSP 1.0 specification. For stores running in WebSphere Commerce 5.4, the JSP templates were required to support the JSP 1.1 specification. When migrating your store toWebSphere Commerce 5.5, you must ensure that the JSP templates in your store comply to the JavaServer Page 1.2 specification created by Sun Microsystems. For information about the JavaServer Page 1.2 specification, refer to Sun Microsystem’s Java Web site at www.java.sun.com.
- The JavaServer Pages Specification 1.2 states that the only language supported is ″java″. Thus, the following page language declaration in your JSPs are no longer valid:

  ```
  <%@ page language="JAVA" %>
  ```

  Note that the WebSphere Commerce Instance Migrator tool (used for payment migration) will convert all occurrences of <%@ page language=″JAVA″ %> to <%@ page language=″java″ %> on your behalf.
- If you are using AbstractAccessBean.getInitContext() to lookup the initial context in your JSPs, it is recommended that you change this to AbstractAccessBean.getInitContext(null,null).
- Remove the `<jsp:root>` and `</jsp:root>` sections in your JSP files since they are only valid if your JSP files are XML documents. If your JSP files are not XML documents (that is they are in JavaServer Pages format), you should remove the `<jsp:root>...</jsp:root>` sections from these JSP files. For more information on this and XML Documents, refer to section ″JSP.5.2 JSP Documents″ of the *JavaServer Pages Specification (Version 1.2)* available from Sun Microsystems and the section entitled ″Updating the WebSphere Commerce 5.4 JSP files″ within the *WebSphere Commerce Migration Guide*.
- Import the java.util.* packages. In WebSphere Application Server, Version 5, JSP files that use the ″Vector″ directive have to explicitly include java.util.Vector package. If you have the following line in your JSPs, your JSPs will not require any changes to work in WebSphere Commerce 5.5:

  ```
  <%@ page import="java.util.*" %>
  ```

For example, if you have migrated the ToolTech sample store, in order the the JSP pages to display properly, you need to add the following line to the top of the JSP page (for example to the top of the CatalogItemAdd.jsp page):

```
<@ page import="java.util.Vector"%>
```

If you do not import the java.util package and you use classes inside of the package, you will need to make the following changes to your JSP files. Common classes used in WebSphere Commerce are as follows:

Enumeration
To import the specific class, use:

```
<%@ page import="java.util.Enumeration" %>
```

Vector To import the specific class, use:

```
<%@ page import="java.util.Vector" %>
```

ResourceBundle
To import the specific class, use:

```
<%@ page import="java.util.ResourceBundle" %>
```

For more information on classes provided with WebSphere Commerce 5.5, see the WebSphere Commerce Production and Development online help.

Environment B post-migration

For Environment B, where the development environment and WebSphere Commerce coexist but do not share assets, do the following:

1. If you are migrating from WebSphere Commerce Studio, Version 5.1 to WebSphere Commerce Studio, Version 5.5, migrate your WebSphere Commerce merchant keys. For more information on migrating your merchant keys, see the ″Migrating your security configuration″ section of the *WebSphere Commerce Migration Guide*.
2. Rename the `*instance*.xml` file to a unique name (for identification purposes only). For example, rename it from `demo.xml` to `demo2.xml`.
3. Modify the `wcs_instances` file in the WebSphere Commerce instances directory (WebSphere\WCS\instances) to point to the new XML file.
4. Edit the `*instance*.xml` file located in the CommerceServerDev\instances\instance\xml directory and update the `name` entry for the database to the new name (for example, `Mall2`). Following is an example:

```
<Database>
<DB DBMSName="DB2"
    RunDB2SG="true"
    SummaryTable="true"
    OraUserID=""
    DBAName=""
    RemoteDB="false"
    DBAPwd=""
    DBServerPort=""
    DBNode=""
    DBHost=""
    DBUserID=""
    DBUserPwd=""
    name="Mall2"
    active="true"
    StagingEnable="false" />
</Database>
```

Part 4. 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 Programming Guide and Tutorials*; for example, the programming guide tutorial illustrates how to extend a controller command, and the migration tutorial explains how to migrate the extended command to WebSphere Studio Application Developer.

Before you try out the tutorials, ensure that you have migrated all the VisualAge for Java store assets 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 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
4. Select all files to import, including sub-folders.
5. Choose the folder \Stores\Web Content to contain your files.

Chapter 6. Migration flow for the tutorials

The following is a high level view of the end-to-end scenario or flow for the tutorials outlined in this guide, using the scenario of three machines: Machine A runs your previous version of WebSphere Commerce for production purposes, Machine B is an empty machine and is for migration, and finally, Machine C is the development machine running VisualAge for Java. The following are the steps required for this scenario:

1. Install required software stack onto Machine B.
2. Run an instance migration from Machine A to B. This may be a remote instance migration.
3. Copy the database from Machine A to Machine B.
4. Run database migration on Machine B.
5. Install WebSphere Studio Application Developer on Machine C.
6. Export and import custom code and enterprise beans from VisualAge for Java to WebSphere Studio Application Developer.
7. Migrate and unit test the custom code and enterprise beans.
8. Deploy (assemble) custom code and enterprise beans into the migrated EAR file resulted from step running the instance migration from Machine A to B. For details on deployment, refer to the *WebSphere Commerce Programming Guide and Tutorials* guide.
9. Deploy (install) the EAR. For details on deployment, refer to the *WebSphere Commerce Programming Guide and Tutorials* guide.
10. Test that all custom code and enterprise beans works appropriately.
11. Shut down production on Machine A.
12. Copy the database from Machine A to Machine B.
13. Run database migration on Machine B.
14. Switch configuration to use Machine B as the production server.
15. Start WebSphere Commerce 5.5 on Machine B.

Chapter 7. Tutorial A: Migrating an extended or customized controller command

The following tutorial walks you through migrating 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. The tutorial within the *WebSphere Commerce Programmer's Guide* for your previous version of WebSphere Commerce illustrated how to extend the existing OrderProcess controller command so that the total bonus points that have been accumulated on the purchase are displayed on the order confirmation page. For this tutorial, you will the newly extended command, OrderProcessCmdBonusImpl, as a model for the controller command migration instructions.

Migrating an enterprise bean

In the same *WebSphere Commerce Programming Guide and Tutorials* 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 a 1.1. JAR, deleting the ExtensionJDBCHelperBean in the WebSphereCommerceServerExtensionData folder, importing the enterprise bean into WebSphere Studio Application Developer, and generating the deploy code.

Exporting your enterprise bean from VisualAge for Java

- If you are migrating from WebSphere Commerce Studio, Version 5.1, you used VisualAge for Java 3.5.3. Do the following:
  1. Export the WCSSampleEntitiesBeansProject as a repository named ejb.dat from VisualAge for Java 3.5.3.
  2. Import this repository ejb.dat to the VisualAge for Java 4.0 environment using the Export Tool for Enterprise Java Beans 1.1 as mentioned in the bullet below. You will need to have installed VisualAge for Java 4.0 for this step. If you are a WebSphere Commerce Studio, Version 5.1 customer migrating to WebSphere Commerce Studio, Version 5.5, contact WebSphere Commerce Support to obtain a copy of VisualAge for Java 4.0. It is recommended that you install VisualAge for Java 4.0 on a single machine as the transition machine for migrating enterprise beans.
  3. Export the WCSSamplesEntity Bean enterprise bean group as an enterprise bean 1.1 JAR.
  4. Import the enterprise bean 1.1 JAR file into WebSphere Studio Application Developer, Version 5.

- If you are migrating from WebSphere Commerce Studio, Version 5.4, you can do this by using the Export Tool for Enterprise Java Beans 1.1. If you are a WebSphere Commerce Studio, Version 5.1 customer migrating to WebSphere Commerce Studio, Version 5.5 and have contacted WebSphere Commerce Support to obtain a copy of VisualAge for Java 4.0, you can use the Export Tool. You must export all the enterprise bean groups containing custom enterprise beans to a single JAR file as follows:
  1. In VisualAge for Java, add the **Export Tool for Enterprise Java Beans 1.1** feature to the workspace if you have not already added it.
  2. On the EJB page of the Workbench, right-click the **WCSSamplesEntity Bean** enterprise bean group and click **Export** > **EJB 1.1 JAR**.
  3. Fill in the fields as necessary. Ensure that the **.java** check box is selected.
  4. Select your database (such as UDB DB2 or Oracle).
  5. Ensure that you select your database.
  6. Click **Finish**.

Deleting the ExtensionJDBCHelperBean in the WebSphereCommerceServerExtensionData folder

1. In WebSphere Studio Application Developer, go to the J2EE perspective.
2. From the J2EE Hierarchy view and expand **EJB Modules**.
3. Expand **Session Beans**.
4. Expand the **WebSphereCommerceServerExtensionsData** folder.
5. Right-click **ExtensionJDBCHelperBean** and click **Delete**.
6. Click **Select All** and click **OK**.

Importing your enterprise bean into WebSphere Studio Application Developer

1. In WebSphere Studio Application Developer, go to the J2EE perspective.
2. Select **File** > **Import** > **EJB JAR file**. Click **Next**.
3. Select your the EJB project called **WebSphereCommerceServerExtensionsData**.
4. Select the EAR project called **WebSphereCommerceServer**.
5. Click **Next** and select all the EJB files from your JAR file.
6. Click **Next** and select all the dependent projects. If you are unsure as to which projects you have a dependency on, select all of them.
7. Click **Finish**.

Editing methods to avoid java.rmi.RemoteException errors

1. In WebSphere Studio Application Developer, go to the J2EE perspective.
2. From the J2EE Navigator view, open **WebSphereCommerceServerExtensionData\ejbModule \com.ibm.commerce.sample.objects\BonusBean.java**

3. Modify the following methods so that they do not throw java.rmi.RemoteException errors:

   ```
   ejbLoad()
   ejbStore()
   unsetEntityContext()
   ```

   For example, change `public void ejbLoad() throws java.rmi.RemoteException` to `public void ejbLoad()`
4. Save your changes.

Changing the access isolation level

1. In WebSphere Studio Application Developer, go to the J2EE perspective.
2. From the J2EE Navigator view, open ejb-jar.xml under \WebSphereCommerceServerExtensionData\ejbModule\META-INF
3. Click **Access**.
4. Select the node **Committed**, which is below **Isolation Level**, and click**Remove**.
5. Click **Add**, select **Repeatable read**, and click **Next**.
6. Select **Bonus** and click **Next**.
7. Expand **Bonus** and select the check box for all home methods (that is, the first icon under **Bonus**).
8. Click **Finish**.
9. Verify that the node **Repeat** now appears under **Isolation Level**.
10. Save your changes.

Generating deploy code

Since you have modified code, you must regenerate its deployed code.

To regenerate the deployed code, do the following:
1. In WebSphere Studio Application Developer, go to the the J2EE Hierarchy view.
2. Expand the **EJB Modules** folder.
3. Right-click **WebSphereCommerceServerExtensionsData** and select **Generate** > **Deploy and RMIC Code**.
4. Select **All EJBs** and click **Finish**.

Migrating a customized JSP file

In the same *WebSphere Commerce Programming Guide and Tutorials* 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\store_directory
4. Select confirmation.jsp to import.
5. Choose the folder \Stores\Web Content to contain your files.

Migrating an extended controller command

To migrate an extended controller command, you must first export the project 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, and click **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 above.
4. Select all .java files in the JAR to import.
5. Choose the folder \WebSphereCommerceServerExtensionsLogic\src to contain your files.

To compile migrated code in the Web project, follow these steps:
1. Open WebSphere Studio Application Developer and switch to the J2EE perspective.
2. From the J2EE Navigator view, right-click **WebSphereCommerceExtensionsLogic**.
3. Select **Properties**.
4. Select **Java Build Path**.

5. Select the **Projects** tab.
6. Select **WebSphereCommerceExtensionsData**.
7. Select the **WebSphereCommerceServerExtensionsLogic** project.
8. Right-click and select **Build Project**.

Testing your new logic in the FashionFlow sample store

Before you can verify your new business logic, you must start your server. To do this, from the Servers view, right-click **WebSphereCommerceServer** and select **Start**.

To verify your new business logic in the FashionFlow sample store that is running in WebSphere Studio Application Developer, do the following:

1. 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://hostname/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.
2. Select and purchase a product.
3. After you have purchased a product, the order confirmation displays the number of bonus points that were earned on the order.

Chapter 8. Tutorial B: Migrating an extended or customized entity bean and task command

The following tutorial walks you through migrating 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 second half of the tutorial in the *WebSphere Commerce Programmer's Guide* for your previous version of WebSphere Commerce illustrated how to customize an existing task command, GetProductContractUnitPriceCmd, and its corresponding interface. As part of this tutorial, you will create new business logic to calculate a discounted price, based upon the customer's balance of bonus points. This calculation is 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

To migrate an extended task command, you must first export the project 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, and click **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 above.
4. Select all .java files in the JAR to import.

5. Choose the folder \WebSphereCommerceServerExtensionsLogic\src to contain your files.

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

1. Open WebSphere Studio Application Developer and switch to the J2EE perspective.
2. From the J2EE Navigator view, open **WebSphereCommerceServerExtensionsLogic\src\com\ibm\commerce \sample\commands\GetNewBaseUnitPriceCmdImpl.java**.
3. Search for `new BigDecimal(dblBonusPrice)` and change it to `new java.math.BigDecimal(dblBonusPrice)`.
4. Save the file.
5. Select the **WebSphereCommerceServerExtensionsLogic** project.
6. Right-click and select **Build Project**.

Migrating an extended data bean

The next step in the tutorial is to migrate the extended data bean, NewProductDataBean.java. To migrate an extended data bean, you must first export the project 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, and click **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 above.
4. Select all .java files in the JAR to import.
5. Choose the folder \WebSphereCommerceServerExtensionsLogic\src to contain your files.

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

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

Migrating an enterprise bean

In the same *WebSphere Commerce Programming Guide and Tutorials* 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 re-create the CMP field to the User entity bean, do the following:

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

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" on page 55.

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.
2. From the J2EE Hierarchy view, expand on **Databases**.
3. Expand **User database schema**, and then **NULLID**.
4. Right-click **Tables** and select **New** > **New Table Definition**. The Table Definition Editor opens.
5. Type `BONUS` for the table name and click **Next**.
6. In the Table Columns page, create the following two columns:
   - memberid - BIGINT, key column (for DB2) or memberid - NUMBER, key column (for Oracle)
   - bonusPoint - INTEGER, nullable
7. Click **Next** twice to go to the Foreign Keys page.
8. 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
9. From **Source Columns**, select **member_Id** and click the **>** button.
10. 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.
2. From the J2EE Hierarchy view, expand **EJB Modules**.
3. Right-click **Member-MemberManagementData EJB module** and select **Open With** > **Mapping Editor**.
4. From the upper left pane (the Enterprise Beans pane), expand **Member** and select **User bean**.
5. From the upper right corner pane, hold down the **Ctrl** key and select the **MEMBER**, **USERS** and **BONUS** tables. All three tables should be highlighted.
6. From the pull-down menu, choose **Mapping** > **Create Mapping**.
7. From the pull-down menu, choose **Mapping** > **Match by Name**.

8. From the Overview window, select **User bean**.
9. Move to the Properties window and expand **Bean to Table Strategy**.
10. Change the **Descrimator Value** from `User` to `U`.
11. Save your EJB mapping (you can press **Ctrl+S** on your keyboard).

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 as described in "Generating deploy code."

Generating deploy code

Since you have modified the code for the User entity bean, you must regenerate its deployed code.

To re-generate the deployed code, do the following:

1. In WebSphere Studio Application Developer, go to the the J2EE perspective.
2. From the J2EE Hierarchy view, expand **EJB Modules**.
3. Right-click **Member-MemberManagementData EJB module** and select **Generate** > **Deploy and RMIC Code**.
4. Select **All EJBs** and click **Finish**.

Migrating a customized JSP file

In the same *WebSphere Commerce Programming Guide and Tutorials* 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. Choose the folder \Stores\Web Content to contain your files.

Testing your new logic in the FashionFlow sample store

Before you can verify your new business logic, you must start your server. To do this, from the Servers view, right-click **WebSphereCommerceServer** and select **Start**.

To verify your new business logic in the FashionFlow sample store that is running in WebSphere Studio Application Developer, do the following:

1. 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://*host_name*/webapp/wcs/stores/servlet/StoreCatalogDisplay?storeId=*store_Id*&catalogId=*catalog_Id*&langId=-1

   where:

   host_name
      The fully qualified host name of your WebSphere Commerce development environment machine

   store_Id
      The identifier for your store.

   catalog_Id
      The identifier for your store's catalog.
2. Click the **Register** link under the **Services** heading and then click **Register** under the **New Customer** heading.
3. Register a new customer using the e-mail address `wctester@wc` and the password `wctester1`.
4. Fill in other fields with test values and click **Submit** and leave the browser open.
5. 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 step 5c on page 82.

   f. Click the **Execute** icon.

6. In the browser, click the **Men's** link to view the men's fashions section of the store.
7. 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 5. Appendixes

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.

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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

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

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

Trademarks

Printed in USA