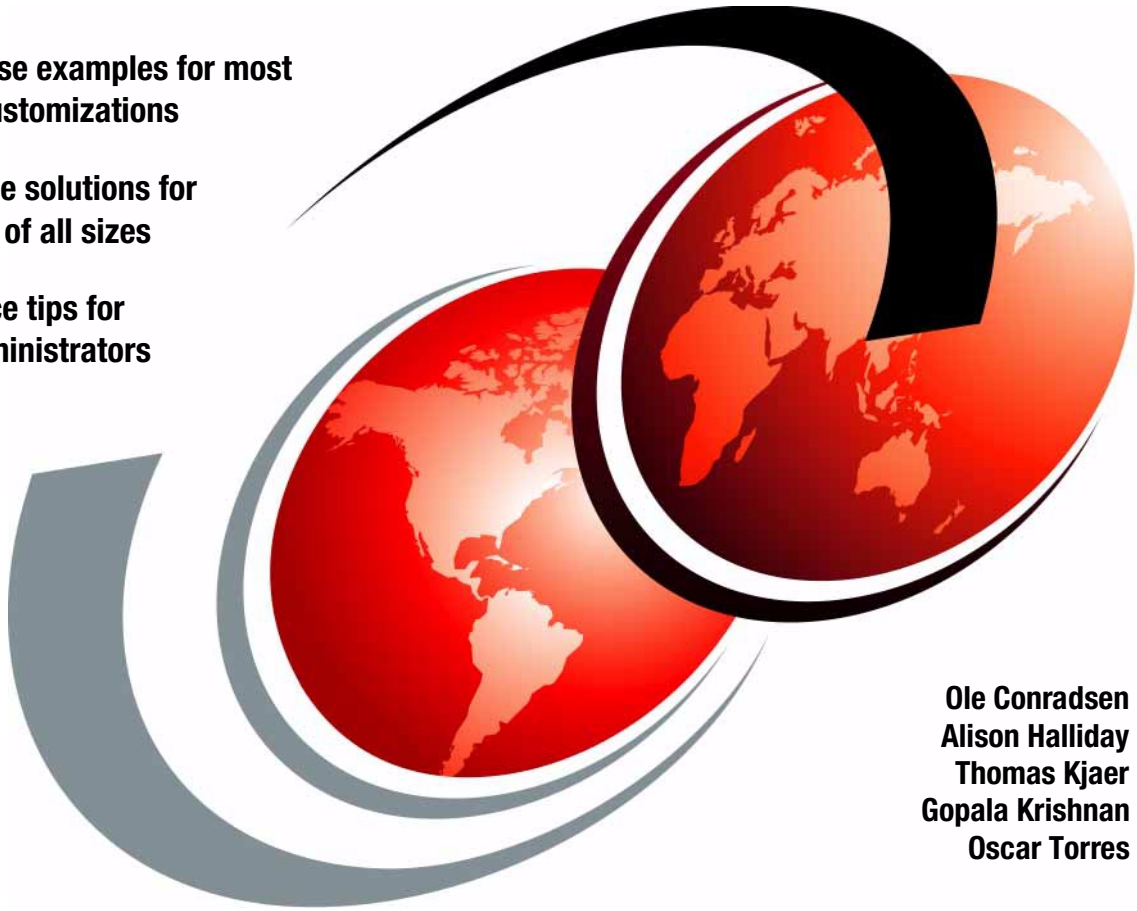
IBM

IBM WebSphere Commerce Suite SPE Customization

- Ready-to-use examples for most common customizations
- E-commerce solutions for businesses of all sizes
- Performance tips for system administrators



Ole Conradsen
Alison Halliday
Thomas Kjaer
Gopala Krishnan
Oscar Torres

Redbooks

ibm.com/redbooks

SG24-5958-00

International Technical Support Organization

IBM WebSphere Commerce Suite SPE Customization

November 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix K, “Special notices” on page 393.

First Edition (November 2000)

This edition applies to WebSphere Commerce Suite, Service Provider Edition (SPE) Version 3, Release 2 for use with AIX and Windows Operating Systems.

This document created or updated on November 30, 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 003 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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

Contents

Figures

Tables

Preface

This redbook contains hints and tips for customizing WebSphere Commerce Suite Service Provider Edition Version 3.2. The purpose of this book is to show the flexibility of the product, offer ideas for customization, and give real ready-to-use working examples with source code and installation instructions. It is our goal to enable the reader to install and configure the product and perform common customizations. The first chapter describes installation on AIX and AIX configuration considerations. These installation tips will improve performance and reliability for WCS on AIX. Other chapters describe how to set up a unique URL for a merchant, customer group, cross link, and shipping customizations. All examples in this redbook are accompanied by source code, which makes it possible for the reader to use the examples instantly. It was our intention that the examples be ready to use, and we supplied enough information for you to change the examples if they do not exactly fit your needs. The source code for all examples is available from the ITSO Web site as described in Appendix J, “Using the additional material” on page 391. The current eddition was updated November 2000 adding examples and correcting others.

The team that wrote this redbook

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

Ole Conradsen is a Project Leader at the International Technical Support Organization, Austin Center. He writes extensively about HACMP, HAGEO, Lotus Domino, and e-business solutions. Before joining the ITSO, he worked on AIX projects in IBM Denmark for 11 years. Ole holds a Master of Science degree in Electrical Engineering from The Technical University of Denmark.

Alison Halliday is an IT-Specialist from IBM Global Services, Sweden. She has four years experience in the Internet and e-business technology areas and has worked with NetCommerce since 1997. Alison holds a Master of Science degree in Computer Science from Queen's University, Belfast, N. Ireland. Her areas of expertise include NetCommerce and WebSphere.

Thomas K. Kjaer is an IT-Specialist in Denmark. He has five years experience with Web technology and three years experience with Net.Commerce. Thomas holds a B.A. in Information Science from Aarhus University, Denmark. He has worked at IBM for two years, and his areas of expertise include the Net.Commerce and WebSphere Application Server product groups.

Venkatachalam Gopala Krishnan is a Software Engineer in IBM Global Services, India. He has two years experience in Web Development at IBM and holds a degree in Mechanical Engineering from Bharathiar University, Coimbatore, India. Gopal is a Sun Certified Java Programmer, and his areas of expertise include Net.Commerce and WebSphere Application Server.

Oscar Humberto Torres MartÌn del Campo is a Project Leader in México. He has two years experience in several fields. He has worked at Telecom & Soft, S.A. de C.V. (an IBM Business Partner) for two years. His areas of expertise include Lotus Domino (development and administration) and IBM e-business solutions with different platforms.

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

Greg Krysa
IBM Toronto Lab

Chris Mann
IBM WebSphere Commerce Suite product manager

Mike Polan
IBM Toronto Lab

Jeff Shiner
IBM Toronto Lab

Daesung Chung
IBM International Technical Support Organization, Austin Center

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 415 to the fax number shown on the form.
- Use the online evaluation form found at http://www.redbooks.ibm.com/
- Send your comments in an Internet note to redbook@us.ibm.com

Chapter 1. Installing WCS Service Provider Edition

This chapter describes how to install WebSphere Commerce Suite (WCS) Service Provider Edition (SPE) for AIX. The chapter is intended for system administrators or anyone else who is responsible for performing similar tasks. This book often refers to the software, WebSphere Commerce Suite, Service Provider Edition, by its short name, WCS SPE.

Note

We decided to keep this chapter as short and precise as possible and only cover the installation of the software that is bundled with the WCS SPE CD. We assume that the reader of this book has knowledge of AIX and has performed some previous installations of AIX. You may refer to the *Installing and Getting Started Guide V3.2*, GC09-2808, for detailed information and step-by-step installation instructions.

To find out about the most recent product information including any late changes, see the README file in the root directory of the WCS SPE CD.

1.1 Products included with WCS SPE

The following products are packaged with WCS SPE:

- WebSphere Commerce Server 3.2
- IBM DB2 UDB 6.1.0.6
- DB2 Text Extenders 6.1
- IBM Net.Data 6.1
- Lotus Domino Go Webserver 4.6.2.61
- IBM Payment Server 1.2.20.3
- Netscape Communicator 4.61 for Windows
- IBM WebSphere Application Server 2.02 and JDK1.1.6

Compatible Web servers

Although Domino Go Webserver Rel. 4.6.2.61 is the Web Server provided with WCS SPE, you can use Netscape Enterprise Server Rel 3.61 as the Web server.

Compatible databases
Although DB2 UDB 6.1.0.6 is the database provided with WCS SPE, you can, alternatively, use Oracle 8.05 or Oracle 8.06 as your database.

If you have DB2 UDB 5.x currently installed, you must upgrade it to DB2 UDB 6.1.2 and migrate your database.

Supported Web browsers
WCS SPE access to the Administrator is supported using Netscape Communicator 4.61 from any Windows NT machine that is on the same network as your WCS SPE machine.

Shoppers can access Web sites by using any of the following Web browsers, all of which have been tested with WCS SPE:

- Any version of Netscape Navigator supported with Netscape Communicator 4.61, including Netscape Navigotor4.04,4.06 and 4.08
- Netscape Navigator 4.0 for machintosh
- Internet Explorer 3, 4, and 5

1.2 Hardware and software requirements

This section explains the hardware and software required to install WCS SPE.

1.2.1 Hardware requirements

You must ensure that you meet the following minimum hardware requirements before installing WCS SPE 3.2:

- To run WCS SPE 3.2, you need a dedicated RISC System/6000 or IBM Power Series family of machines (RS/6000 43PModel 150 or higher is recommended) with the following minimums:
  - 200 MHz processor
  - Minimum of 128 MB of random access memory (RAM)
  - Minimum of 900 MB of free disk space
  - CD-ROM drive
  - Graphics-capable monitor
- To access the WebSphere Commerce server Administrator, a personal computer capable of running Windows NT, Windows 98, or Windows 95 with a graphics-capable monitor, color depth of at least 256 colors, and a mouse or other pointing device is needed.

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

1.2.2 Software requirements

You must ensure that you meet the following minimum software requirements before installing WebSphere Commerce server on an AIX system:

- Ensure that you have AIX 4.3.2 or 4.3.3 on your WCS SPE machine. If you are using AIX 4.3.2, you must have the following fixes applied:
  - You must apply PTF U453695 for CSet++ miscellaneous runtime fixes.
  - Ensure that the C runtime xlC.rte is at least at Version 3.6.4.1. To verify this, type the following at a shell prompt:

    ```
    lslpp -l | grep xlC.rte
    ```

    If the `xlC.rte` is below the required level, apply PTF U463530. This PTF will update the `xlC.rte` to version 3.6.6.0 and is included on the WCS SPE CD.
- Ensure that you have the following installed on the machine(s) you will use to access the WCS SPE Administrator:
  - Windows NT, Windows 98, or Windows 95
  - Netscape Communicator 4.61 or newer version (a copy is provided on the WCS SPE CD in the /netscape directory).

1.3 Disk space and paging space requirements

This section describes the required disk space and paging space for the installation of WCS SPE. If disk space is not sufficient, the installation would stop at midpoint and return *There is not enough space in the file system*. Ensure that you have sufficient disk space and paging space before starting the installation.

1.3.1 Verifying disk space

You must have the following minimum amount of free space:

- 400,000 blocks of free space (at 512 KB per block) in the /home directory. This is required for DB2. If there is not enough space in the /home directory, the DB2 install may fail or encounter difficulties. Depending on the size of your database, you may require even more free space.
- 2,000,000 blocks of free space (at 512 KB per block) in the /usr directory

- Create a file system called /installer with 400,000 blocks of free space (at 512 KB per block). This may be used to store temporary files during installation. This will help prevent the / (root) file system from getting filled up because filling up the root file system would jeopardize the complete system. It is not recommended that you fill in the root system at any cost.

To determine whether you have enough space, type `df` on an AIX command line, and look for information about the /home, /usr, and root directories. If free space is less than required space, use the smit tool to increase free space.

1.3.2 Verifying paging space

You must have at least 128 MB of paging space. If you are using Netscape, you need to add another 20 MB of paging space. But, in general it is recommended to have twice as much paging space as physical memory, that is, 256 MB if you have 128 MB of memory in the system. For systems with more than 1 GB of physical memory, the paging space requirements are lower.

To determine whether you have enough paging space, type `lsps -a` from a command line. Add up the sizes of all the active paging spaces. If the total is less than 128 MB, use the smit tool to increase paging space.

1.4 Common installation steps

This section describes the installation procedures most common to all software bundled with the WCS SPE CD. You may have to refer to this section for the installation tips, and, if there is any difference for a particular software, it is mentioned in the relevant subsections. Perform the following steps:

1. Login is as user ID `root`
2. If necessary, mount the CD.
3. Change to the CD-ROM install directory. This varies for the respective software to be installed and is mentioned in the relevant subsection.
4. On the command line, type `smit install_all`.
5. In the Input device / directory for software field, type `./` and click **OK** or press **Enter**.
6. Click on the **List** button for software to install. Using the cursor movement keys and the space bar, make your selection.
7. Depending on the software you are installing, highlight and select the needed part of the software as explained in the relevant subsection.

8. When you have made all selections, click **OK** or Press **Enter** to close the list.
9. A confirmation message appears. Click **OK** or Press **Enter**. The command Status window appears indicating the installation of the selected product has started. The installation is completed when the Command field at the top of the window changes from *Running* to *OK*.
10. When the installation is complete, scroll to the Installation Summary section at the bottom of the listing. In the RESULT column, you should see either *Success* or *Already Installed* next to the name of each component. If you do not see this, correct the problem and attempt the installation again.
11. Press **F12** (Exit).
12. Return to the root directory.
13. Unmount and remove the CD.

1.4.1 Installation procedures for Domino Go 4.6.2.61

This section explains how to install Domino Go Webserver 4.6.2.61.

The following steps are the installation procedure for Domino Go Webserver 4.6.2.61. Refer to Section 1.4, “Common installation steps” on page 4.

1. Change to the CD-ROM install directory for Lotus Domino Go, and type:
   `cd /CDROM_dir/usr/sys/inst.images`
2. Domino Go specific installation instructions in smit or smitty are

Highlight the following packages that apply to your copy of the Domino Go Webserver:

- NetQ.* - This is required if you want to perform searches on the Domino Go Webserver documentation.

Select one of the following:

- If you are installing the North American version of Domino Go Webserver, select **gskru301**.
- If you are installing the Export version of Domino Go Webserver, select **gskre301**.
- internet_server.base - The server, administrator, and documentation.
- internet_server.loc.lang - Where lang is the code page value for your locale, such as en_US for American English.
- internet_server.msg.lang - The message catalog, where lang is the code page value for your locale, such as en_US for American English.

- internet_server.security.common - The common security files.
- internet_server.security.us_secure - The security files for use in Canada and the United States.
- internet_server.security.export - The security files for use outside Canada and the United States.

Note

Do not install the internet_server.java.* packages. You will install another Java servlet engine during the WebSphere Application Server installation process later.

3. Follow the remaining steps per the *Common Installation steps* and complete the installation.

1.4.2 Enabling SSL on Domino Go Webserver for testing

The following section describes the steps involved in creating a security key ring for testing. As you perform these steps, your browser may display security messages. Carefully review the information in each message and decide how to proceed.

Creating a security ring for testing

To create a security ring for testing, do the following:

1. Ensure Domino Go Webserver is started by typing `lssrc -s httpd` on a command line. If the status is inactive, start the Webserver.
2. On your NT machine, start your Web browser; disable and clear all disk and memory caching, and disable all proxy servers.
3. To access the Web server's home page, type the following on your browser:

   `http://host_name/path`

   where `path` is the name of your Web server's home page if you have done any customization on the Webserver.
4. Click **CONFIGURATION AND ADMINISTRATION FORMS**.
5. When prompted, type your Webserver administrator user ID and password and click **OK**. The default administration user ID is webadmin, and the password is webibm.

   If the authorization fails because of non availability of the webadmin user during installation, change to the /usr/lpp/internet/server_root/protect directory from the command line and type

```
htadm -adduser webadmin.passwd webadmin webibm admin
```

This will create the user ID, webadmin.

6. On the Configuration and Administration Forms page under Security, select **Create Keys**.
7. On the Create Key and Request Certificate form, select certificate type **Other**, and click **Apply**.
8. On the Other Certificate form, in the Key name field, type `testnetc`. In the Key ring field, type `/usr/lpp/internet/server_root/testnetc.kyr`.
9. Change the Size field to the highest setting that is available.
10. Under Key Ring Password, in both Password fields, type a key ring password of your choice. You will need this password later to change the default key in the key ring and to receive certificates into that key ring.
11. Check the **Automatic login** box.
12. Complete the fields under **Distinguished Name**. For Server name, use the fully-qualified name of your Websphere Commerce server.
13. Under Mail to, select **Don't mail**.
14. Under Save Copy, in the Save certificate request to file field, type `/usr/lpp/internet/server_root/testnetc.txt` and click **Apply**. You should see a confirmation page indicating that you have successfully created your public-private key pair and certificate request. If you get a message indicating an error instead, retry these steps. The error message should indicate the problem that was encountered.

Setting your test key ring as the current key ring

To make the Web server use your test key ring, do the following:

1. Return to the Configuration and Administration Forms page by clicking **Configuration Page** at the bottom of the confirmation page.
2. Under Security, click **Security Confirmation**.
3. On the Security Configuration form, under Default key rings, select **/usr/lpp/internet/server_root/testnetc.kyr**.
4. Select **Set selected key ring as current key ring**.
5. Click **Apply**. A page appears confirming that the security configuration changes have been made.

Receiving and testing the test key ring certificate

To receive and test your key ring certificate, do the following:

1. Return to the Configuration and Administration Forms page by clicking **Configuration Page** at the bottom of the confirmation page.
2. On the Configuration page, under Security, click **Receive Certificate**.
3. On the Receive Certificate form, in the Name of file containing certificate field, type `/usr/lpp/internet/server_root/testnetc.txt`.
4. In the Key ring field, type `/usr/lpp/internet/server_root/testnetc.kyr`.
5. In the Key ring password field, type the password that you used to create the key ring.
6. Click **Apply**. You should see a confirmation page that indicates that the certificate was successfully received.
7. Return to the Configuration page.
8. Under Security, click **Key Management**.
9. On the Key Management form, in the Key Ring Password field, type the password that you used to create the key ring.
10. Select **Designate Trusted Root Keys**, and click **Apply**.
11. On the Designate Trusted Root Keys form, under Keys, select **testnetc** from the list.
12. Click **Apply**. You should see a confirmation page that indicates the operation was successful.
13. Stop and start the Web server by typing `stopsrc -s httpd` and `startsrc -s httpd` from AIX command line.

1.4.3 Verifying the installation

To verify the installation of the Lotus Domino Go Webserver, do the following:

1. On your Windows machine, open your Web browser.
2. Enter the following URL: `http://host_name`
3. If you can access the Web server home page, your Web server is running.

To verify that the SSL is working, do the following:

1. Enter the following URL in the Web browser (be sure it is https, not http): `https://host_name`
2. If you can access the Web server home page, SSL has been enabled.

1.5 Installing IBM DB2 UDB 6.1.0.6

This chapter explains how to install IBM DB2 Universal Database 6.1.0.6 (DB2 UDB). To complete the steps in this chapter you will require the IBM DB2 Universal Database 6.1.0.6 CD.

1.5.1 Installation procedure

The following is the installation procedure for IBM DB2 UDB 6.1.0.6. Refer to the common installation steps for general installation instructions and the instructions unique to UDB in the following section. Perform the following steps:

1. To install DB2, type the following in an AIX command line
   `./db2setup`
   The db2setup program calls another program that ensures that you have the necessary PTFs on your system. If you do not, the program asks you if you want them to be automatically installed. Type `y`, to automatically install them, or `n`, to exit the db2setup program. If you type y, the PTFs are automatically installed, and you are prompted to reboot and rerun db2setup. If there are no required PTFs missing, you may proceed further.
2. If you are requested to reboot, type the following on an AIX command line:
   `bosboot -a`
   `shutdown -Fr`
3. The db2setup program scans your system for information about your current configuration and displays a window called Install DB2 V6.1. Highlight your selections and press the space bar or press **Enter** to select them as follows:
   a. If you intend to run a Commerce Service Provider database on this machine, select **DB2 UDB Enterprise Edition**.
   b. If you intend to run all your Commerce Service Provider databases on remote machines, select **DB2 Administration Client**.
   c. If you want DB2 messages displayed in a language other than English, select **Customize** beside DB2 Product Messages to open the DB2 Messages window. Then, highlight your language code, press the **space bar**, highlight **OK**, and press **Enter**.
   d. If you want to install the DB2 publications in HTML format in a language other than English, select **Customize** beside DB2 Product Library and press **Enter** to open the DB2 Product Library window. Then, highlight your language code, press the space bar, highlight **OK**, and press **Enter**.

4. When you have made selections, highlight **OK** and press **Enter**.
5. The Create DB2 services window appears. Select **Create a DB2 Instance** and press Enter. The DB2 Instance sub-window appears. Complete the following details:
   a. User Name - Type the DB2 instance ID that you want to use. (The instance ID, db2inst1, is used as an example in this book.) The instance ID must meet the following criteria:
      - It cannot be more than eight characters in length.
      - It can contain only the characters A to Z, a to z, 0 to 9, @, #, $, and _.
      - It cannot begin with an underscore (_).
      - It cannot be any of the following in upper, lower, or mixed case: USERS, ADMINS, GUESTS, PUBLIC, or LOCAL.
      - It cannot begin with any of the following in upper, lower, or mixed case: IBM, SQL, or SYS.
   b. Group Name - Type a group name that you are not currently using for any other user ID. This group will automatically become the system administration group for the DB2 instance, and it will be given administration authority.
   c. Password - Type a password that meets the following crietria:
      - It cannot be more than eight characters in length.
      - It can contain only the characters A to Z, a to z, 0 to 9, @, #, $, and _.
      - It cannot begin with an underscore (_).
        Verify Password - Type the same password again.
        Accept the defaults for all other fields, and press **Enter**
6. The Fenced User window is displayed. Accept all the defaults by highlighting **OK** and pressing **Enter**.
7. A Notice window is displayed advising you that a system generated password will be used. Highlight **OK** and press **Enter**.
8. The Create DB2 Services window is displayed. Highlight **OK** and press **Enter.**
9. Ignore the warning message that indicates that the Administration Server is not created by highlighting **OK** and pressing **Enter**.
10. A summary report appears listing the components that will be installed. Highlight **Continue** and press **Enter**.

11. A warning appears advising you that this is your last chance to stop the installation. Highlight **OK** and press **Enter**.
12. The db2setup program installs your components and creates your instance ID in the group you specified. Depending on the speed of your processor, this can take up to 15 minutes. When it completes, a Notice window informs you whether it was successful or not. Highlight **OK** and press **Enter**.
13. Scan the Status Report to ensure that all components were installed successfully and that the DB2 instance ID was created successfully. Highlight **OK** and press **Enter.**
14. To close the DB2 Installer window, highlight **Close** and press **Enter**.
15. Ignore the message that indicates that the Administration Server is not created by highlighting **OK** and pressing **Enter**.
16. To confirm that you want to exit DB2 Installer, highlight **OK** and press **Enter**.
17. Log on to the instance ID you just created by typing `su - db2inst1` (or the instance ID you typed earlier).
18. Edit the .profile file using a text editor as follows:

    Append `/usr/lib` to the LIBPATH environment variable after the PATH statement if it does not appear there already. It should be followed by a line that exports the `LIBPATH`. The syntax of these lines should be:

    ```
    LIBPATH=/usr/lib
    export LIBPATH
    ```
19. To ensure that there are no errors, run the .profile file by typing the following on an AIX command line:

    ```
    . .profile
    ```
20. Type `exit` to return to user ID root.
21. In a AIX command window, type `smit users` or `smitty users`.
22. Select **Change/show characteristics of a user**.
23. Enter the name of the user in the Enter name of the user field
24. Ensure that the Initial Program field is set to /usr/bin/ksh.

1.5.2 Verifying the installation

To verify the installation, perform the following steps:

1. From an AIX command window, log in as `db2inst1`.
2. type `db2`. This takes you to the db2 = > prompt.

3. Type `create database dbname` where `dbname` is any database name you want to create
4. Type `list database directory` to ensure that the database has been created.
5. Type `connect to dbname`. This connects the database we created.
6. Type `create table MyTable (fieldone Integer, field char(20))`.
7. Type `list tables` to see the created table.
8. Type `quit` to return to the AIX command line.
9. Type `db2 connect reset` to reset from the database.

1.6 Installing IBM DB2 Text Extenders

The DB2 Extenders consist of the DB2 Image Extender, the DB2 Audio Extender, the DB2 Video Extender, and the DB2 Text Extender components. WCS SPE requires DB2 Text Extender to be installed. The following are the installation procedures for IBM DB2 Text Extenders. Refer to the common installation steps for general installation instructions and the instructions unique to Text Extender in the following section. Perform the following steps:

1. The DB2 extenders install directory is /CDROM_dir/db2ext/aix.
2. To select the DB2 Extenders for installation, select the components from the following list depending on whether you are installing the DB2 Extenders server or client:
   - `db2ext_06_01.client` - DB2 Extenders client component
   - `db2ext_06_01.server.nonee` - DB2 Extender server component
   - `db2ext_06_01.doc.language` - DB2 Extenders online information where language corresponds to your language, for example, En_US for U.S. English
   - `db2tx_06_01.dic.laguage` - DB2 Text extenders dictionaries to install on the client
3. After installation, to configure the DB2 Text Extender, do the following:
   a. Establish the DB2 Extenders instance
      1. Ensure you are a active root user
      2. Change to DB2 instance directory by typing
         `cd /usr/lpp/db2ext_06_1/instance`
      3. Run the DMBINSTANCE command as follows:
         `./dmbinstance <instance_id>`

Running `dmbinstance` creates the /u/instanceid/dmb directory. Do not create additional files or directories under the /u/instanceid/dmb directory. These files are lost if the instance is deleted.
As dmbinstance runs, you are prompted to confirm the following:

- The request to make a DB2 owner instanced (if it is not already a DB2 instance owner).
- The request to create an instance for the DB2 Image, Audio, and Video Extenders (IAV) Server. Respond **YES.**

b. Establish the DB2 Extenders instance environment:

1. Log in as `instance id`.
2. The dmbprofile script is provided for the Korn shell to establish the DB2 Extenders environment. The dmbprofile shell scripts contain statements that update paths in your operating system and set environment variables. To establish the DB2 Extenders instance environment, include the DB2 Extenders instance profile into the login profile for the instance id user ID. Add the following statements to the end of the login profile for instance id:

```
LANG=locale
export LANG
. dmb/dmbprofile
```

To ensure that there are no errors, run the.profile script by typing the following on an AIX command line

```
. .profile
```

3. Type `exit` and press **Enter** to return to user ID root.

For more information on installing and configuring DB2 Extenders as well as additional topics, such as creating and managing multiple extender instances, see the file, install.txt, in the db2ext/aixeee/language directory of the CD-ROM.

1.7 Installing IBM Net.Data 6.1

This section describes how to install Net.Data 6.1. If you are installing or reinstalling Net.Data after installing Websphere Commerce Server, ensure that you back up your db2www.ini file in the /usr/lpp/internet/server_root directory before installing or reinstalling Net.Data. (Net.Data installs its own version of this file.) Once you have completed installing or reinstalling Net.Data, you can replace the Net.Data version of the file with your backup version of the file.

1.7.1 Installation procedures

The following are the installation procedures for IBM Net.Data 6.1. Refer to Section 1.4, “Common installation steps” on page 4, for general installation instructions and the instructions unique to Net.Data in the following section

1. Net.Data installation program is found in the directory, /CDDROM_dir/code/aix.
2. Highlight the following packages in smit or smiity
   - Net.Data
   - Net.Data.msg.locale - locale corresponds to your language. For example, en_US for U.S. English.
3. After installation, test your installation by doing the following:
   a. Open the browser on your NT machine.
   b. Type in the URL
      `http://host_name/cgi-bin/db2www/hello.d2w/report`
   c. If *Hello World!* appears in the browser, the installation is OK.

1.8 Installing JDK1.1.6 and WebSphere Application Server

This section describes how to install WebSphere Application Server 2.02 Standard Edition and JDK 1.1.6. To complete the steps in this chapter, you will need the WebSphere Application Server 2.02 Standard Edition CD.

WebSphere Application Server is a requirement for using Product Advisor. If you are not going to be using Product Advisor, the installation of WebSphere Application Server is optional.

1.8.1 Checklist

To ensure that you successfully complete the steps in this chapter, you must meet the following requirements:

1. You must have a Web server (either Domino Go Webserver, Netscape Enterprise Server, or Domino Web Server), Net.Data, and a relational database management system (either DB2 or Oracle) installed before beginning the steps in this chapter. During the WebSphere installation, you will be prompted to install a JDK. Do not install the JDK from this prompt because you will install the correct JDK in Section 1.8.2, Installing JDK 1.1.6, below.
2. AIX 4.3.3 includes JDK 1.1.8. If you have this level of AIX installed, uninstall this version of the JDK using SMIT or SMITTY by selecting all

JDK packages and removing them. Then, install JDK 1.1.6 using the steps in this chapter.

3. You must have graphical monitor or terminal to install WebSphere Application Server. (Telnet from the NT machine would not be sufficient).

1.8.2 Installing JDK 1.1.6

To install JDK1.1.6, perform the common installation steps and the following steps, which are unique to JDK 1.1.6:

1. The install directory of JDK is CDROM_dir/AIX/JDK.
2. To select JDK1.1.6 for installation, highlight **Java.adt** and **Java.rte**.
3. After installing JDK1.1.6, type `java -fullversion` from the command line. The following response appears:
   ```
   java full version "JDK 1.1.6 IBM build a116-19991124 (JIT enabled:
   jitc)"
   ```
4. By default, the JDK gets installed in the /usr/jdk_base directory
5. To verify the installation, write a Hello World Java program and execute it from the command line.

1.8.3 WebSphere Application Server installation procedures

To install WebSphere Application Server, do the common installation steps and the following steps unique to WebSphere Application Server. We recommend that you use the same AIX console and machine to start the installation rather than doing a telnet from an NT system and installing the WebSphere because the installation program is GUI based.

1. Change to the cd /CDROM_dir/AIX directory.
2. Enter the `./install.sh` command to start the installation program.
3. The installation program will display your current JDK level and path. If asked if you wish to install the JDK 1.1.6 from the WebSphere Application Server CD, enter `n`. You should have already installed the correct JDK as outlined in the previous section. Ensure that you do not install the version of JDK that WebSphere Application Server prompts you to install as the preferred level of JDK. That version of the JDK does not contain the patches that the Product Advisor requires.
4. The installation program will prompt you for the JAVA_HOME path. Enter the path where the JAVA_HOME is located. By default, this is /usr/jdk_base.
5. A Welcome window appears. Click **Next** to continue.

6. If the License Agreement window appears, review the License Agreement. To accept the terms, click **Yes**.
7. On the Choose Destination Location window, either accept the default installation path, /usr/WebSphere/AppServer, or enter a new installation path by clicking **Browse**. When the correct path is specified, click **Next** to continue.
8. The next window allows you to select which WebSphere Application Server components you wish to install. Select the following:

- In the left-hand selection box, select the following:

  a. **Application Server**

  b. **Documentation** (optional)

  c. **AppServer Administrator**

- With Application server highlighted in the right-hand selection box, select the following:

  a. **Servlet Engine** (base)

  b. The appropriate plug for the Web server you are using as follows:

     - If you are using Netscape Enterprise Server, select **Netscape V3.51**.
     - If you are using Domino Go Webserver, select **Go Webserver V4.6.1**.

9. When you have selected the components to install, click **Next**.
10. On the next window, you are prompted to confirm that you want to begin copying files. If you want to begin the installation, click **OK**. To return to the Component and Subcomponent window, click **Cancel**.
11. When the installation is complete, the Setup Complete window appears. If you want to review the readme file, ensure that the checkbox is not selected. Click **Finish**
12. To test the installation, stop and start your Domino Go Webserver by issuing the following commands from your AIX command line.

```
stopsrc -s httpd
startscr -s httpd
```

13. Type the following URL from your browser in your NT machine. Use admin as the default user ID and password:

```
http://host_name:9527/
```

14. Highlight **Servlets** and click **Configuration**.

15. Click **snoop** under **Servlet Names** and click **Load**. The snoop servlet will now be loaded.
16. Open a new browser and type the following URL:
    `http://host_name/servlet/snoop`
17. If you get the results of the snoop servlet, your installation is OK.

1.9 Installing IBM Payment Server

To install IBM Payment Server, do the common installation steps and the following steps unique to IBM Payment Server:

1. The install directory for IBM Payment Server is/CDROM_dir/payment_server.
2. In the software to install field in smitty, type `all`.
3. Follow the steps in Section 1.4, “Common installation steps” on page 4, and finish the installation.

1.10 Installing WebSphere Commerce Service Provider

To install WebSphere Commerce Server, perform the steps described in Section 1.4, “Common installation steps” on page 4, and the following steps unique to WebSphere Commerce Server:

Note

Ensure that you have installed your Web server, your database, Net.Data, JDK, WebSphere Application Server, and Payment Server (if applicable) before you install WebSphere Commerce Server.

1. If you are using DB2, ensure that it has been started.
2. Verify the level of the xlC run-time library by typing the following:
   `lslpp -l | grep xlC.rte`
3. The install directory for WebSphere Commerce Server is/CDROM_dir/wcsspe.
4. Highlight the NetCommerce3.CHS package in smit or smitty and commence the installation.
5. After installation is complete, change to the /usr/lpp/NetCommerce3/server/bin directory in the AIX command line and type the following:

`./start_admin_server`
This starts the administrator server.

6. On your browser, from your NT machine, type the following URL `http://host_name:4444/`. Use webadmin and webibm as the user ID and password.
7. Create a new instance by clicking on **New**.
8. Accept all the default settings, enter the database password, and click **OK**.
9. It takes around 15 to 20 minutes for the instance and the database to be created - depending on your machine.
10. Once the database is created and populated, once more, type in the URL:
`http://host_name:4444/`
and start the Net.Commerce instance by clicking **Start**.
11. Once the instance has been started, a message in the status bar will read *Server Status : Active*.
12. Type in the following URL to verify the functionality of the WebSphere Commerce Server:
`http://host_name/cspsite`
If you get the default CSP home site, the installation of WCS SPE is complete.

1.11 Configuring WCS SPE

After installing WCS SPE, it is a good idea to check some basic settings that can be changed for each product instead of just staying with the default settings. WCS SPE consists of several independent products, and, by changing a few parameters, you can easily improve your system's performance, reliability, and availability.

Since WCS SPE consists of many different products, tuning your environment for the best performance depends on factors, such as what hardware is being used and the number of merchants and shoppers, all of which we will not be able to cover in this book. Instead, we will point out a few key areas where performance improvements can be achieved without much effort.

When you have installed WCS SPE and it is running, we suggest that you take a look at the following topics:

- Net.Commerce settings
- LDGW setting

When running a hosted environment, such as the WCS SPE, you should be aware that each merchant will use the merchant tool to update and publish their site. This means that merchant activity could have an overall performance impact to your site, which you must be able to handle. In general, it is a good idea to encourage your merchants to update their stores only during off peak hours to allow the best possible performance during the day.

One of the best ways to improve performance on a Net.Commerce system is to make use of the dynamic page cache facility, which will cache category and product pages. By caching these pages, Net.Commerce only has to query the database for categories or products the first time any user requests one of these pages. The Net.Commerce version used in WCS SPE v3.2 supports two different levels of caching: Basic and advanced. However, the caching in Net.Commerce will affect all stores in the environment; so, before enabling the caching, you should consider how it will affect the category and product pages if some merchants have customized them in any way.

1.11.1 Net.Commerce settings

WCE SPE v3.2 is based on Net.Commerce v3.2, which you can also tune in many ways.

Server processes

When you have installed WCS SPE and you are ready to create the initial WCS instance, it is possible to specify how many server processes Net.Commerce is starting. The number of server processes determines how many simultaneous transactions your system will be able to handle, and the default value is two. Increasing this number will increase the load on your server, but it will also allow more transactions to be handled simultaneously.

As a starting point, you should start with four processes per CPU and then monitor your system for a period watching CPU load, paging, and CPU wait activity using the AIX commands `iostat` and `vmstat`. As long as CPU load is less than 80 percent, paging is low, and the CPU wait is less than 5 percent, you can increase the number of server processes. For the best settings, you should monitor your system during peak hours when the traffic load is at its highest level.

Changing the number of server processes can be done in two ways:

- You can change the value by using the Configuration Manager:

  `http://HOSTNAME:4444/`

Selecting your instance settings, you will find the tab named **Commerce Suite Server** where you can specify the number of server processes.

- You can edit the configuration file by hand.

  a. Locate and open the configuration file in your editor. The configuration file should be named something like the following:

     `/usr/lpp/NetCommerce3/instance/<instname>/config/ncommerce.conf`

  b. In the configuration file, locate the line that reads:

     `PROCESSES 2`

     The value depends on what was previously specified. See Figure 1.

```
SERVICE_NAME_PREFIX ncITSO
EXEC /usr/lpp/NetCommerce3/bin/server
MS_HOSTNAME f50.itsc.austin.ibm.com
DBNAME ITSO
DBINST db2inst1
DBOWNER db2inst1
DBPASS bCCT2m6cuzw=
DB_HOST
DB_NODE
DB_REMOTE OFF
PROCESSES 2
MERCHANT_KEY QunGZnDUqUDK7yW0cEnk38vOvgkAO1Ym
```

Figure 1. Editing the PROCESS variable in the configuration file

  c. Change the value `2` to the required number of processes.

  d. Save the file and exit the editor.

In both cases, you will need to restart WCS in order to make the change become effective.

Transactions count

In combination with the PROCESS value, you should monitor the setting of MS_TRANS_COUNT. In Net.Commerce, the servers, as specified by the PROCESS directive, recycle when it reaches the number of transactions specified by this value.

You can increase the value of this setting until your system starts to get memory paging. The recommended value for MS_TRANS_COUNT is between 250 and 1000 and should not be set higher than 5000 for this version of Net.Commerce.

To change the value, you will have to edit the configuration file for Net.Commerce. To do this, perform the following steps:

1. Locate and open the configuration file in your editor. The configuration file should be named something like the following:

   ```
   /usr/lpp/NetCommerce3/instance/<instname>/config/ncommerce.conf
   ```

2. In the configuration file, locate the line that reads:

   ```
   MS_TRANS_COUNT 1000
   ```

   The value depends on what was previously specified.

```
 MTOOL_NUM_CON 5
MTOOL_MAX_WAIT 10
NC_JRE_PATH /usr/jdk_base/bin
NC_CLASSPATH /usr/jdk_base/lib/classes.zip:/usr/lpp/db2_06_01/java/db2ja
.zip:/usr/lpp/NetCommerce3/CHS/CHS.jar:/usr/lpp/NetCommerce3/CHS/Nc_chs.jar
ERR_NOTIFICATION ON
ORDER_NOTIFICATION ON
ADMIN_ERR_MSG_TEMPLATE errmsgadmin.tpl
TEMPLATE_PATH /usr/lpp/NetCommerce3/templates/en_US
USERTRAFFIC_LOG 1
MS_TRANS_COUNT 1000
```

Figure 2. Editing the MS_TRANS_COUNT variable in the configuration file

3. Change the value `1000` to the required value.
4. Save the file and exit the editor.
5. Restart WCS to make the change become effective.

To determine which value to use, over a period of time, monitor the available memory on your system for a given value of MS_TRANS_COUNT. If memory becomes low, performance will be poor or you will get the error message, Server Not Responding, frequently; so, decrease the value.

Logfiles

For a production environment, all Net.Commerce logs should be set to level 0 in order to minimize the disk-write activity that occurs when writing to log files. However, if, for some reason, you need logging to some degree, make sure that the log files are located on a separate disk to maximize parallel I/O activity.

To change the log level or the location of the log files, look for the following lines in the configuration file:

```
MS_LOGPATH /usr/lpp/NetCommerce3/instance/mser/logs
MS_LOGLEVEL 2
```

You can set the log level individually for the following components in WCS:

- **server controller** - srvrctrl.conf

- **server process** - ncommerce.conf
- **scheduler** - scheduler.conf
- **payment -** pay_back.conf, pay_cyber.conf and pay_etill.conf

You may also want to clean up your Net.Data macro path since, by default, it also references the demomall and ncsample directories.The Net.Data path is specified by the MACRO_PATH and INCLUDE_PATH directives in the db2www.ini file, which can be found in the /usr/lpp/internet/server_root/pub/ directory.

1.11.2 LDGW settings

The Lotus Domino Go Web Server serves all requests submitted by the browser and has many settings that can be adjusted for better performance.

Log files
The Web server logs all kinds of traffic including errors. It will generate several log files to record this information. By default, the Web server generates the following logfiles: access, agent, referrer, httpd-error, CGI-error, and servlet logs. Because a lot of disk writing will occur, you should review which logs to turn on or off before going into production with your system. If there is no need for the logged data, all logs except the error logs should be turned off.

As with Net.Commerce logs, you should place the Web server logs on a different disk to maximize parallel I/O activity because the log files for the Web server will be some of the fastest growing files.

You can change the log settings on the LDGWs *Configuration and Administration Forms* in the *Logging and Reporting* section.

Pass directives
The pass directives in the Web server specify which directories can be accessed with a browser. Before going into production, you should review these settings and only keep the minimum required setting. For example, you could remove every reference to the demomall, the grocery mall, and so on. Keeping the pass directives to a minimum can also improve the performance of the Web server since the server takes the requested URL and processes it through the list of mapping rules. Processing ends when the request is accepted, rejected, or redirected to another server. If you have many rules, you should consider the order of the rules in the list because processing starts from the top of the list and continues down. You can change the pass directives on LDGWs *Configuration and Administration Forms* in the *Request Processing* section, and then select **Request Routing**.

Cache

LDGW also received a cache feature, which must not be confused with the Net.Commerce cache. The caching in LDGW can cache static files allowing for quick response to static files, which includes HTML files, image files, or other kinds of files whose content will never change.

However, one drawback with the LDGW caching is that every file you want the Web server to cache must be explicitly specified in the configuration file before the Web server starts. Changes to the files of cached files will require a restart of the Web server.

You can change which files to cache on LDGWs *Configuration and Administration Forms* in the *System Management* section, and then select **Caching**. For further information about WCS SPE configuration and tuning, look for the following PDF documents on the WCS Web site:

`http://www-4.ibm.com/software/webservers/commerce/servers/lit-tech-general.html`

or

`http://www.software.ibm.com/commerce/net.commerce`

- Net.Commerce Configuration and Tuning Updated 29 February, 2000
- Webserver Configuration and Tuning Updated 29 February, 2000
- DB2 Configuration Parameters Updated 29 February, 2000
- Net.Commerce Configuration Planning Guide Updated 14 February, 2000

Chapter 2. Configuration of the server environment

The Websphere Commerce Suite (WCS) is a powerful product composed of several IBM products. It includes Domino Go Webserver, DB2 Universal Database, and more.

The Application Services Provider (ASP) can make use of the features in these products and customize them, thus, merchants are able to offer added value to their customers and be more competitive.

In the first part of this chapter, we will talk about how the ASP can configure the Domino Go Webserver to obtain some benefits.

Another very important subject is where and in which directories all the information files for each merchant on the server reside. This is very important for the ASP because having good data organization is the key to preventing any sharing problems between information for different merchants. In the second part of this chapter, we will talk about how the WCS is organized in the server. Using this information will make it very easy to get an idea of where and how to modify, protect, and even achieve customization.

2.1 Unique URL

The objective with this section is to show how each merchant can get a unique URL such that the mall or CSP URL is not shown on the Web browser. The URL of the mall must be hidden, even in the dynamic pages, to give each shop a more unique identification.

In many cases, merchants want to be contacted via the Internet by their own domain. For example, a merchant who creates the store, *myfirstIBMstore*, in a WCS SPE environment with the hostname, *thehostname*, must type `http://thehostname/myfirstIBMstore` to see the store.

Often, the merchant wants or already has the Internet domain name www.myfirstIBMstore.com. The merchant wants customers to get access to the store from the mall address, but also from the Web site: `http://www.myfirstIBMstore.com`.

The WCS SPE has been designed and developed to serve not just one merchant but a group of merchants. It is, therefore, very important for the ASP to have this information and the knowledge to offer this service to the merchants.

2.1.1 Running the server with multiple IP addresses or virtual hosts

The ASP may want to use one server to provide Web service for multiple merchants. For example, you might have two merchants (merchant A and merchant B), both of whom want to make information about their companies and/or stores available on the World Wide Web (WWW). You might want to put both Web sites on the same machine if the expected number of requests for information is not large enough to justify a separate machine for each customer.

With the Domino Go Webserver, you can use multiple IP addresses, virtual hosts, or both to provide multiple Web sites on one server.

2.1.1.1 Multiple IP addresses

To use multiple IP addresses, your server must be installed on a machine with multiple network connections, have one or more adapters, or use IP alias where each adapter has more than one IP address.

If you have only one network connection and run two instances of the server on the same machine, only one server has the benefit of using the default port number. Requests to the other server would have to include a port number.

If you are running WCS under AIX, be aware that AIX has a method of assigning an alias name to the same network adapter. In this way, you can have several Web sites in the same server with only one network adapter.

For more information about the alias names under AIX, see the *AIX Version 3.2 Commands Reference, Volume 3*, GC23-2367.

If your machine has two or more adapters or network connections, you can run just one instance of the server and assign each merchant a different IP address. For each IP address, you must define a different host name. For example, merchant A could be `www.merchantA.com` on IP address 9.67.106.79, and merchant B could be `www.merchantB.org` on IP address 9.83.100.45. You could then configure the server to serve a different set of information depending on the IP address the request comes in on. Because the server can accept requests from the default port of each network connection, requests to either host name could be directed to the default port number.

If your server has multiple IP addresses, you can associate a specific secure sockets layer (SSL) key ring label for each individual IP address. For example, if the server has two different IP addresses configured (125.25.116.87 and 125.25.116.89) with Domain Name Server (DNS) entries of `www.Mall.mycom` and `www.SuperShopper.com`, respectively, the keyfile database

could contain two separate sets of keys and certificates. Each hostname/IP address combination would have its own corresponding certificate.

To take advantage of SSL support for multiple IP addresses, you must assign each host name a different IP address and then define a specific key ring label per IP address.

2.1.1.2 Virtual hosts

With virtual hosts, no additional hardware is required, and you can save IP addresses. However, clients must support HTTP 1.1 or HTTP 1.0 with 1.1 Extensions. The reason for this is that Version 1.0 of HTTP does not support multiple URLs for a single TCP connection. The problem has been solved with Version 1.1 of HTTP.

With virtual hosts, you can run just one instance of the server and assign each merchant to a different virtual host. In the DNS, you define your hosts and associate them with the IP address of your server. You can then configure the server to serve a different set of information depending on the host for which the request is made. Requests do not require a port number.

Note

Another advantage, if you chose the *Virtual Host* option, you don't need a SSL certificate for each merchant. It means that the same certificate is used for the mall and every store in it. The cost of maintaining the certificate can be shared among all the merchants in the mall.

2.1.1.3 Setting up your server

Setting up your server to use multiple IP addresses or virtual hosts is very similar. For multiple IP addresses, you will need to specify the IP address a request comes in on, and, for virtual hosts, you will need to specify the host name for which a request is made.

You configure the server to serve different information for each merchant by indicating that certain parts of your configuration apply only to requests coming in on certain addresses or for certain hosts. You can configure three key parts of your server so that requests are processed differently based on the IP address they come in on or the host name in the URL: Welcome Pages, Mapping rules, or Access control.

Welcome pages

You can specify different sets of file names to use as welcome pages depending on the address on which the request comes in or the host name in

the URL. The file names you define as welcome pages determine how the server responds to requests that do not contain a specific file name.

For example, you might want to specify that homeA.html is a welcome page only for requests received on 9.67.106.79 or for hostA, and homeB.html is a welcome page only for requests received on address 9.83.100.45 or for hostB.

From the Configuration and Administration forms page, you can configure your list of welcome pages by clicking on **Initial Page**. From the Initial Page form, click the help icon for information on defining welcome pages and how to associate a welcome page file name with an IP address or a host name.

Alternatively, if you are editing the configuration file, you can add an IP-address or host name at the end of a Welcome directive to associate a welcome page file name with an IP address or a host name. Select **Directories and Welcome Page => Set viewing options** and view the details and the description of the Welcome directive.

Mapping rules

You can specify a different set of mapping rules for the server to use depending on the address a request comes in on or the host name in a URL. Mapping rules map a request to a physical file on the server and determine whether the server processes a request.

For example, you might want to specify that a request beginning /cgi-bin/ received on IP address 9.67.106.79 or for hostA should be mapped to the /merchantA/cgi/ directory, and the same request received on IP address 9.83.100.45 or for hostB should be mapped to the /merchantB/cgi/ directory.

From the Configuration and Administration forms page, you can configure your mapping rules by clicking on Request Routing. From the Request Routing form, click the help icon for information on how to use mapping rules and how to associate a mapping rule with an IP address or a host name.

Alternatively, if you are editing the configuration file, you can add an IP-address or hostname at the end of Exec, Fail, Map, Pass, and Redirect directives to associate the directive with an IP address or a host name. Select **Resource mapping => Redirect URLs** for details and the description of these directives.

Access control

You can activate different protection rules for a request based on the address the request comes in on or the host name in a URL. Protection rules are

defined in protection setups and determine how your server controls access to files and programs.

For example, you might want to specify that a request beginning with /cgi-bin/ and received on address 9.67.106.79 is protected by the rules in a protection setup named PROT-A, and the same request received on 9.83.100.45 is protected by the rules in a protection setup named PROT-B.

From the Configuration and Administration forms page, you can configure how protection is activated by clicking on **Document Protection**. From the Document Protection form, click the help icon for information on protecting documents and how to associate protection with an IP address or a host name.

Alternatively, if you are editing the configuration file, you can add an IP-address or a hostname at the end of the DefProt and Protect directives to associate the directive with an IP address or a host name. For details, see the description of these directives in "Access control - Set up access control for the server".

For our purposes, it is strongly recommended to use the Virtual Host method with the customization of the Welcome Pages.

2.1.1.4 Example

An ASP wants to install the WCS in one server (AIX or NT), and he or she wants to use the power of the Domino Go Webserver for static pages and another application.

He or she wants each merchant to have his or her own unique URL that can be accessed by customers on the World Wide Web. Besides this, he or she wants all the merchants to be able to be accessed in the mall.

What can we do?

For this example, we have the intervention of three persons:

- The ASP administrator who owns the hardware (server, hubs, routers, network cards, and so on) and software and has all the administrator attributes for all the sites.
- The merchant who owns a store and pays the ASP for the Hosting Service and has administrator attributes just for his or her store.
- The customer who buys products and services from the merchant on the Internet using a computer and a Web browser and his or her credit card.

Scenario
We installed a WCS in one server with the hostname, www.themall.com..

Once we were sure that the WCS was working properly, we installed the Merchant Tool. With the Merchant Tool, we created three stores:

- FREYTOR
- JAJAJA
- RISITAS

Those stores can be accessed without any problems by typing http://www.themall.com/<storename> ; so, we now have one mall with three stores.

Now, the owner of the store, FREYTOR, wants to have a unique URL. It means that he or she wants his or her customers to be able to access the store by typing the IP address for the store or http://www.freytor.com beside the mall name, http://www.themall.com/freytor.

Moreover, the merchants want a unique identity, that is, they want to hide the contents of the status bar that appear at the bottom of the browser and hide any other information or status lines that refer to the mall.

Now, we must decide which solution to offer this merchant, the Multiple IP solution or the Virtual Host solution.

For this example, we have chosen to use the Virtual Host solution because it has several advantages in the typical environment for a WCS SPE installation.

The Virtual Host solution is inexpensive to install and maintain because you do not need to buy or install additional hardware. It is also easy because you do not need to do any changes to your server configuration and it only takes a few steps to solve the problem.

Another advantage of using this method is that you do not need to buy a new SSL certificate for every store in the mall. This means that all the stores use one common certificate, the SSL certificate for the mall.

In this example, we assume that the merchant already bought a URL and had the domain name registered in the Network Information Center (NIC).

First, we will set up the Domain Name Server (DNS) to map the URL www.freytor.com to the IP address of the mall, www.themall.com..

When the merchant creates a new store, the store has the status *New*. And the store is not activated in the mall before the status is changed by the ASP

Note

All the stores created with the WCS can have any of the next four statuses:

New.- The store has been created but the Merchant can't open it.
Close.- The store is available to the Merchant, but the customers can't buy.
Open.- The store is open and the Customer can see the catalog and buy.
Lock.- The ASP locks the store and the Merchant can't customize or use it.

After creation, the merchant will contact the ASP because the ASP is the only one who has access permission to update the store status from New to Close. Later, the merchant can update from Close to Open status with the Merchant Tool.

To update the store status from New to Close using the Net.Commerce administration tool (ncadmin), we must perform the following steps:

1. Get in the administration tool by entering the following URL in the browser:
   `http://<hostname>/ncadmin`

   See Figure 3.

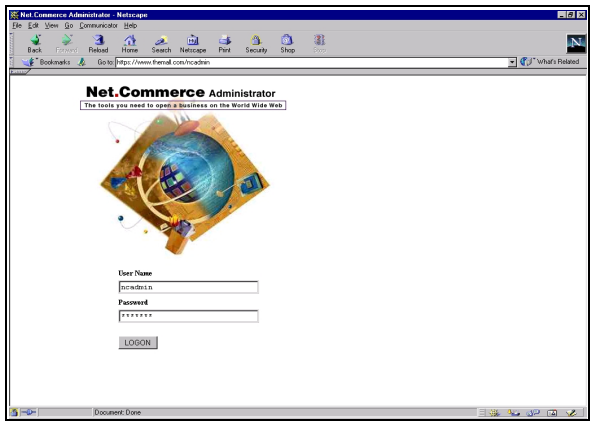


Figure 3. Ncadmin login screen

2. Type the Username and the Password, and then click on the *LOGON* button.

3. On the left menu bar, choose the **Site Manager** option as shown in Figure 4.

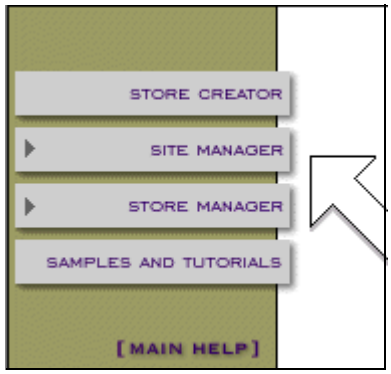


Figure 4. The Site Manager button

4. Inside the Site Manager option, we can see a new menu bar. In this menu bar, we have a new section with the name *CHS*.

   This section is only to manage the stores created under the WCS.

Note

Don't try to setup any store created under WCS with other options than those that appear under the label *CHS*.

This options are only for those stores created with the *Store Creator* under the Net.Commerce schema.

From this menu bar, chose **Store Records** as shown in Figure 5 on page 32.

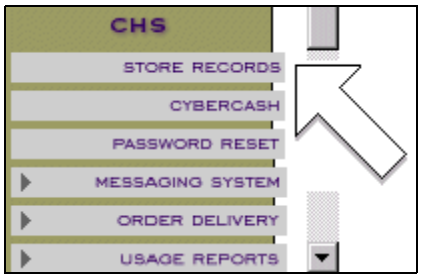


Figure 5. Choose Store Records to get information on stores

5. On the screen appears a form with all the fields left blank. To search for stores, click on **Search** button. Next, as shown in Figure 6, there a list of all the stores created with WCS will appear.

   Select the store to update (in our case, FREYTOR)

Store Number	Store Name	Currency	State	Domain Name	Path
317	Everything Green	USD	Open		everythinggreen
831	FREYTOR	USD	Open		FREYTOR

Figure 6. The list of stores in the bottom frame

   When you have selected the desired store, the fields in the form are filled with the store data.

6. One of the fields is Store Status; so, we must click on the field to chose one of the four statuses available as shown in Figure 7.

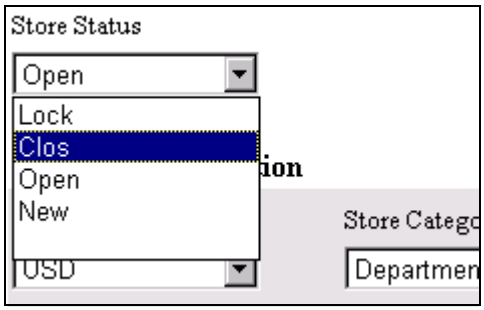


Figure 7. The Store Status field

   So, we choose the Close status and click on the Upgrade button to save the changes.

   Now, the store can be opened and published by the merchant.

   At this point, we can see the store published on the mall. If we type `http://themall.com`, under the All Store Categories option of the MALL DIRECTORY menu, we can see the merchants store (in this case, FREYTOR).

   The next part of the customization consists of creating a *Welcome Page* for our merchant store.

This simple welcome page must be created under the path `/usr/lpp/internet/server_root/pub/` for AIX and `x:\IBM\www\HTML\` for NT, where x: is the logical unit where you install the WCS.

The page consists of three lines of Java script code. In this case, the page named freytor.html of our example looks like the one shown in Figure 8.

```
<script language="JavaScript">
location.href = "/freytor/";
</script>
```

Figure 8. JAVA code for frytor.html

Now, we have a welcome page that refers directly to the merchant store first page; so, we must customize the Domino Go Webserver to be able to serve the information of the merchant by just typing the URL.

To do this, perform the following steps:

1. Type the URL for `http://www.themall.com/frntpage.html`. See Figure 9 on page 34.

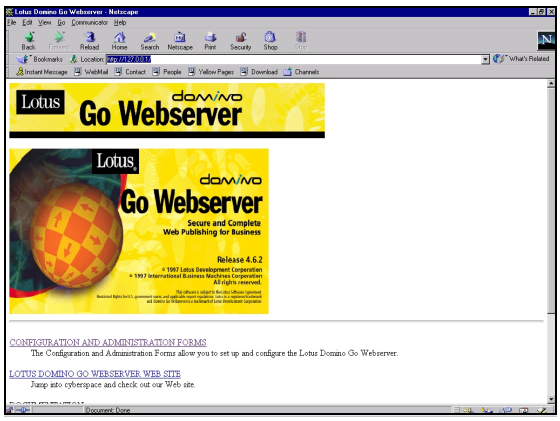


Figure 9. The Domino Go Webserver start

2. Then, select the link **CONFIGURATION AND ADMINISTRATION FORMS**, and a new screen appear asking for your User Name and the Password as shown in Figure 10.

Figure 10. The Username and Password screen

By default, the User Name is *webadmin*, while the Password is *webibm.*

Once the *Username* and the *Password* have been accepted, the whole screen changes.

3. Now, select the **Initial Page** option under the Directories and Welcome Page menu

   Once we do this, the screen will show you a list like the one in Figure 11 on page 35.

Index	File name	Server IP address or host name
Example:	*letsgo.html*	*9.83.**
1	freytor.html	www.freytor.com
2	cspmall.html	www.themall.com
3	Welcome.html	
4	welcome.html	
5	index.html	
6	jajaja.html	www.jujuju.com
7	risitas.html	www.risitas.com
8	Frntpage.html	

Figure 11. The welcome page file list

This list is the welcome page file names database. You can see, for example, that the file name, freytor.html, will appear only for requests from the host name `www.freytor.com`, and so on.

Under the list, you can find some fields and radio buttons as shown in Figure 12. Those are our tools to use if we want to insert, replace, and/or remove a welcome page.

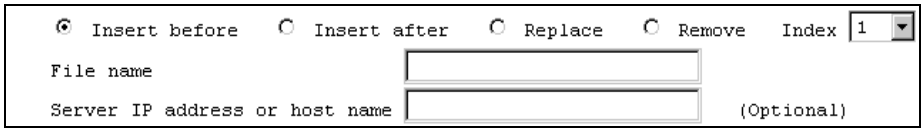


Figure 12. The welcome page file list tools

Inserting a new welcome page

Perform the following steps to insert a new welcome page:

1. Select either **Insert before** or **Insert after**.
2. Select an index number.

   Your choices in steps 1 and 2 indicate the position you want the item to have in the list. For example, if you select Insert before and Index 2, the item will be second in the list. If you select Insert after and Index 4, the item will be fifth in the list.
3. Enter the new file name in the File name field.
4. Optionally, specify the server IP address or host name to associate with the welcome page.
5. Click **Apply** to update your server with the changes you made to the form, or click **Reset** to return to the values that were on the form before you made the changes.

Replacing a welcome page

Perform the following steps to replace a welcome page:

1. Select **Replace**.
2. Select the index number of the item you want to replace.
3. Enter the file name of a new welcome page in the File name field.
4. Optionally, specify the server IP address or host name to associate with the welcome page.
5. Click **Apply** to update your server with the changes you made to the form, or click **Reset** to return to the values that were on the form before you made the changes.

Removing a welcome page

Perform the following steps to remove a welcome page:

1. Select **Remove**.
2. Select the index number of the item you want to remove.

3. Click **Apply** to update your server with the changes you made to the form, or Click **Reset** to return to the values that were on the form before you made the changes.

   Once we insert, replace, and/or remove a row, the Domino Go Webserver presents a confirmation screen with two buttons: The Configuration Page button and the Restart Server button.

   To make more changes to the *welcome page file names* list, select **Configuration Page**.

   To invoke the changes, select **Restart Server**. This causes the server to read the updated configuration file and restart without fully shutting down.

Note

These steps must be followed for every new store. If the merchant don't want a unique URL, then the access to the store must be set to `http://<mall>/<storename>`

At this point, it is possible for the customers to get access to the merchant's store by typing the unique URL (in this case, `http://www.freytor.com`).

This does not mean that access to the store via the mall's URL has been lost. Besides, if the customer types the URL, `http://<mall>/<store>` (for our example, `http://www.themall.com/freytor`), he or she can access the store normally.

The final step is to hide the status bar that appears at the bottom of the browser. When you move the mouse over a link, the address of the link appears in the status line. If the merchant wants to hide the fact that his or her shop is running as part of the mall, the merchant does not want to show the full address in the status bar. Instead, the merchant wants to show a message, such as *Welcome to my store*.To make this happen, we need to type a few Java script lines in the index.html file.

Two variables have an influence on the status bar: window.defaultStatus and window.status. The first one is the contents of the status bar when there is nothing else to put on the status bar. It persists unless the contents of the browser change. The second variable holds the message in the status bar when something changes. The typical use of the second one follows:

```
<body onLoad="window.defaultStatus='Welcome to my store.'">
```

Using those variables, we create a very short Java function to add on the /usr/lpp/internet/server_root/pub/<storename>/index.html file (for AIX) or the

X:\IBM\www\HTML\<storename>\index.html file (for NT) as shown in Figure 13 on page 38.

```
<script language="JavaScript">
     var hellotext="You are now on www.freytor.com. Your 24 hours a day open store"
     var thetext=""
     var started=false
     var step=0
     var times=1

     function welcometext()
     {
       times--
       if (times==0)
       {
         if (started==false)
         {
           started = true;
           window.status = hellotext;
           setTimeout("anim()",1);
         }
         thetext = hellotext;
       }
     }

     function showstatustext(txt)
     {
       thetext = txt;
       setTimeout("welcometext()",4000)
       times++
     }

     function anim()
     {
       step++
       if (step==11) {step=1}
       if (step==1) {window.status='>=======' +thetext+'=======<'}
       if (step==2) {window.status='=>======' +thetext+'======<='}
       if (step==3) {window.status='>=>=====' +thetext+'=====<=<'}
       if (step==4) {window.status='=>=>====' +thetext+'====<=<='}
       if (step==5) {window.status='==>=>===' +thetext+'===<=<=='}
       if (step==7) {window.status='===>=>==' +thetext+'==<=<==='}
       if (step==8) {window.status='====>=>=' +thetext+'=<=<===='}
       if (step==9) {window.status='=====>=>' +thetext+'<=<====='}
       if (step==10) {window.status='======>=' +thetext+'=<======'}
       if (step==11) {window.status='=======>' +thetext+'<======='}
       setTimeout("anim()",200);
     }
 </script>
```

Figure 13. The Java function welcome text()

This code must be invoked on the BODY statement of the /usr/lpp/internet/server_root/pub/<storename>/index.html file (for AIX) or the

X:\IBM\www\HTML\<storename>\index.html file (for NT) as shown in Figure 14 on page 39.

```
<HEAD>
<FRAMESET ROWS=78,* BORDER=0 ONLOAD=setContentsURL();welcometext();>
  <FRAME NAME=BANNER
```

Figure 14. How to invoke the function welcome text

At this point, we have a store that can be accessed with the unique URL, by typing the complete path (`http://<mall>/<store>`), or by choosing it in the mall directory, and we have a Java script function that shows a message instead of the URL of any link.

Note

This steps must be followed to customize any store you want to be accessed by the unique URL and want to hide the contents of the status bar.

2.2 Shop directory structure

One of the most important subjects for any ASP administrator is where and how is the information organized.

Because WCS is a shared environment, it is very important for every ASP to have good organization of the space in the server, and it is even more important when the ASP has several stores owned by several merchants.

With good organization of the files, we will have more control over the space assigned to each merchant, and we will be able to prevent any problem with the server.

Note

In this case, we are talking about WCS running under AIX. In other cases, such as AS/400, Windows NT, and so on, three are minor differences, but, essentially, the structure is the same.

When the product is installed, by default, the installation process creates a structure for each product.

When the Domino Go Webserver is installed, the path, `/usr/lpp/internet/server_root/pub`, is created. It means that all the information beginning from this path can be published on the Internet.

When a new store is created, the WCS creates a sub-directory named after the store under this path. For example, if we create a store named *mystore*, the sub-directory, mystore, will be added to the path, and all HTML files related to this store will be created under it as shown in Figure 15.

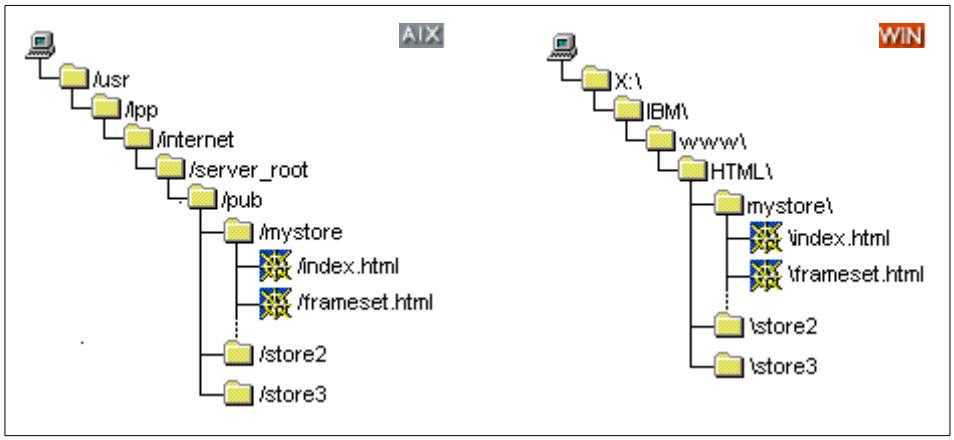


Figure 15. The directory and file structure of the Domino Go Webserver

In Figure 13 on page 38, we showed graphically where the WCS creates sub-directories and where we can find required files.

Something similar happens when Net.Commerce is installed. The Net.Commerce installation path is /usr/lpp/NetCommerce3/macro/en_US, and, in this sub-directory, the WCS creates a new subdirectory under this path for each new store.The name of the sub-directory is the code number corresponding to the store on the DB2 database of this store as shown in Figure 16 on page 41.

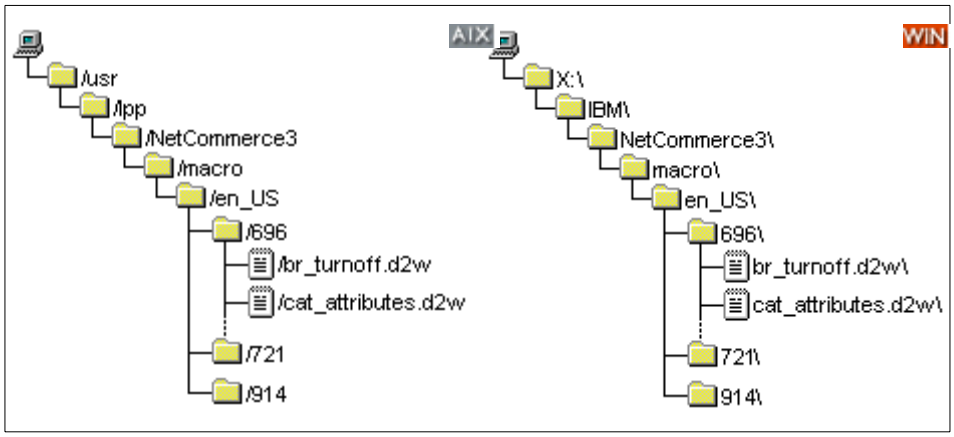


Figure 16. The directory and file structure of the Net.Commerce

As you can see in Figure 15 on page 40, the structure of files and directories of the Net.Commerce (one of the most important components of WCS) is very similar to the structure of the Web Server. Also, in Net.commerce, every store has its own sub-directory and files, and it doesn't matter if the server has many stores or only one of them. If one of the merchants makes a change or customizes anything in his or her own store, the changes will not be reflected in other stores. In addition to these files and subdirectories, there exists another subdirectory structure and file sets that are common to all stores in the mall. If you make changes to these common files, all the stores in the mall will show the changes. In Figure 17 on page 42, we show graphically where in the file structure these common files are, both for AIX and for Windows NT 4.0 Server.

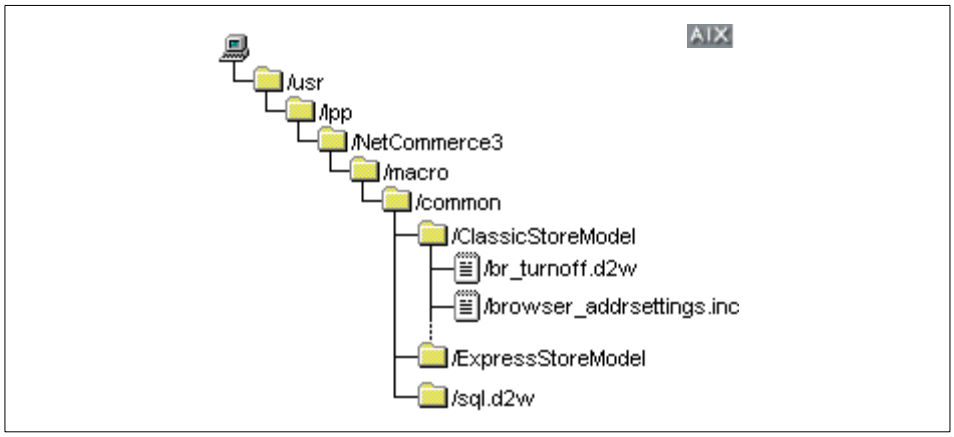


Figure 17. The common files and directories on AIX

Figure 18 shows the common files and directories on Windows NT 4.0.

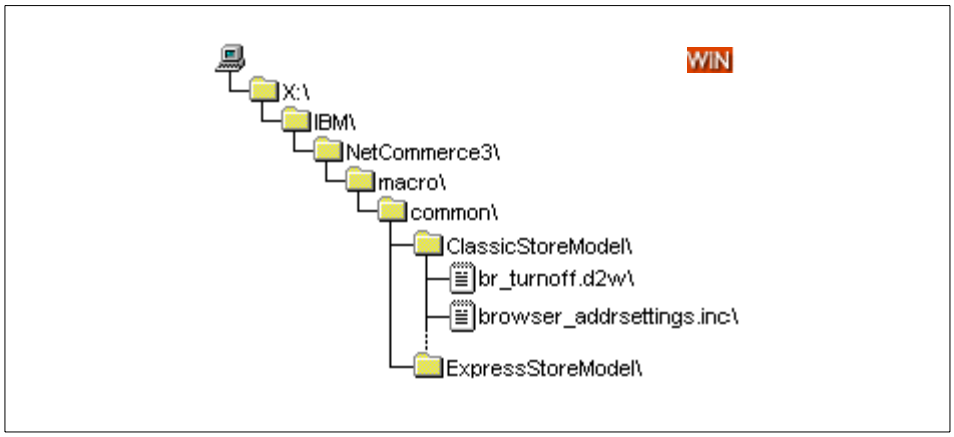


Figure 18. The common files and directories on Windows NT 4.0

Chapter 3. WCS SPE product overview

The WebSphere Commerce Suite, Service Provider Edition is a complex product built upon a number of other products that are very powerful in their own right. In this chapter, we will present an overview of how these components are integrated. Then, we will continue with a more detailed description of each, how they interact in WCS SPE, and what functionalities they address. Lastly, we will discuss the skills and tools required to customize them. This is intended to give a clear picture of WCS SPE to ISPs and to help understand the product and how it may be customized.

3.1 WCS SPE architecture

WCS SPE is a comprehensive set of integrated software components that, together, form the hosting server. Figure 19 illustrates these components.

.

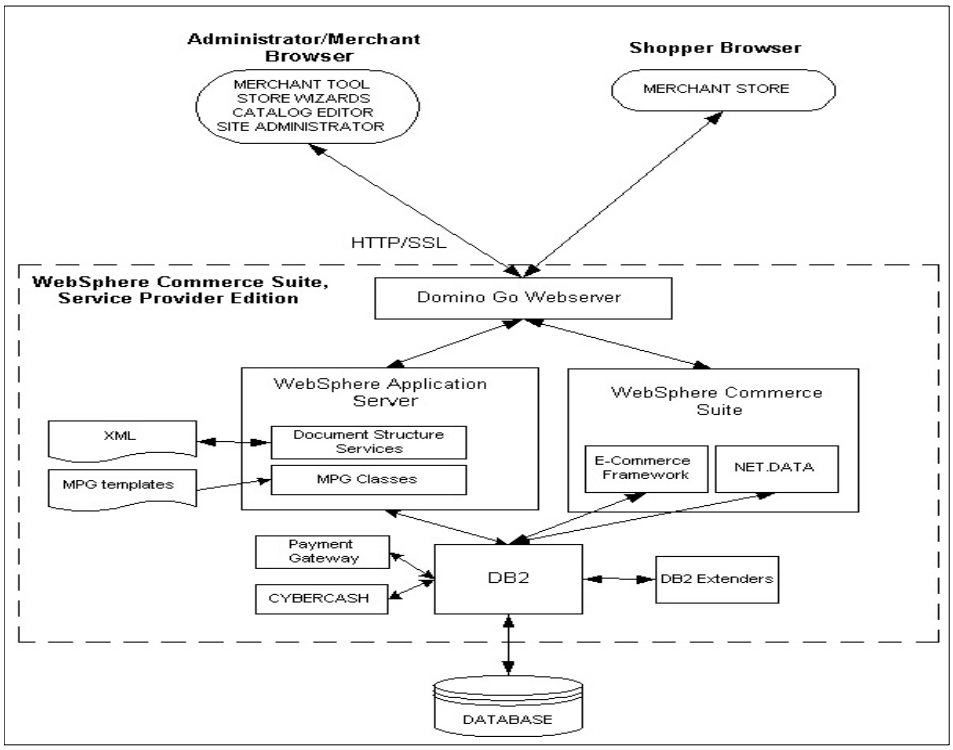


Figure 19. WCS SPE overview

3.1.1 Overview of components

This section goes through the principle components of WCS SPE with functional descriptions and explains its role in the overall WCS SPE product and what tools or skill bases are needed to customize it.

3.1.1.1 DB2

The database is the backbone behind any successful e-business application; therefore, it is also true of WCS SPE. DB2 is a multimedia Web-ready relational database management system for business intelligence and transaction processing. In the case of WCS SPE, it is used to store the merchant's profile, customer data, product portfolio, and details of the order. Most transactions activated by the customer, such as adding a product to the shopping basket, are stored in DB2. This allows for secure shopping and data consistency. DB2 is an advanced product with many performance tuning features and advanced searching tools, such as DB2 Extender. There is an extensive amount of information on DB2. A good starting point is, perhaps, the IBM DB2 homepage at the following Web site:

`http://www.software.ibm.com/data/db2/`

3.1.1.2 Domino Go Webserver

The Domino Go Webserver is a secure, high-performance, transaction-ready Web server supporting a range of Internet standards, such as HTTP 1.0, HTTP 1.1, SSL, and SSI. This product is also a fundamental building block for the WCS SPE and comes fully configured after its installation. Its main function is to redirect the requests from the user to the appropriate service on the host machine, for example, to the Net.Commerce server director or to the Websphere Application Server. It has the customizable functionalities expected of a high-level Webserver and demands the skills typical of a Web master. Further information can be found at the following Web site:

`http://<your_hostname>/Docs/drul0mst.htm`

3.1.1.3 Net.Data

Net.Data enables Internet and intranet access to relational database data, which will then be presented through the customer's browser. This gives Web sites the ability to present dynamic pages to the customer, that is, pages that can be changed depending on certain conditions. For example, if the customer is a discount shopper, they will be presented with the same product page as an ordinary shopper but with their special price. This price difference is dynamically generated. Net.Data works through a CGI program that interprets macro files. These files, typically, contain a mixture of Net.Data

syntax, SQL calls, and HTML tags, which the CGI interprets, resulting in an HTTP response back to the browser.

In WCS SPE, Net.Data is an integral part of the application and is invoked principally for display purposes, for example, a product/category page or a shopping basket. It is a relatively simple but versatile and robust scripting language with a large scope for customizing the presentation of a merchant's store. The skills required are Net.Data programming knowledge and SQL.

Further information and downloadable manuals can be obtained at the following Web site:

`http://www.software.ibm.com/data/net.data/`

3.1.1.4 Websphere Application Server

The WebSphere Application Server (WAS) is a Java application server designed for the management and deployment of Web applications. It is installed on a host Webserver that redirects certain defined requests to it. The WAS then uses the host machine's Java run-time environment to execute the Java programs that make up the Web applications. In WCS SPE, these applications consist of Java Server Pages (JSPs) and Java servlets and are used mostly in the merchant tool and catalog tool. The servlets work in conjunction with XML files (see Section 3.1.1.5, "XML" on page 45) and MPG template files (see Section 3.1.1.6, "Multi Purpose code Generation language (MPG)" on page 47). Principally, Java skills are required to work with the WAS.

There is a Web-based administrator using the following URL:

`http://<your_hostname>:9527/`

For further information on the WAS product, you can look at the following:

- `http://www.software.ibm.com/webservers/appserv/`
- The redbook, *WebSphere Application Servers: Standard and Advanced Editions*, SG24-5460

3.1.1.5 XML

Extensible Markup Language (XML) is used to represent structured data in a text format according to a set of rules or guidelines, that is, platform- and application-independent. Both the structure and content of the data are represented through a tag-based format that is easily generated and read by a computer. These tags, or elements, convey the meaning of the data they contain, not how they should be presented to the customer. Here is an

example of an XML file from WCS SPE that is used to define a sample store that is set up during installation. It has been edited slightly for simplicity:

```
<store storeName="WCSSPE Service Store">
<pages fileName="homePage"
gui="/servlet/MerchantAdmin?DISPLAY=CTnchs.page_editor.pageGUIs.HomePage"
htmlGenerator="html/nchs/page_editor/simplePages/homePage.html"
htmlHeader="html/nchs/page_editor/htmlPageHeader.html"
htmlTrailer="html/nchs/page_editor/htmlPageTrailer.html">
</pages>
</store>
```

which relates in C++ terms.

```
store.storeName
store->pages.fileName
store->pages.gui
store->pages.htmlGenerator
store->pages.htmlHeader
store->pages.htmlTrailer
```

It is an extensible markup language derived from SGML that gives you the flexibility to define your own set of tags and allow third parties to use and understand them.

The Websphere Application Server product contains document structure services that allow the generation and manipulation of XML files using Java. In WCS SPE, the data required to define the store creator wizard screens or the merchant tool, for example, is saved in XML files. Java servlets are used to parse these and create HTML output to the browser, based on their contents. The WCS SPE also generates xmls, for example, when defining a new store, a default file is created, settings.xml, which is then built upon to produce the layout for the new store.

XML configuration files in NCHS

Figure 20 on page 47 shows the relationship among NCHS configuration files.

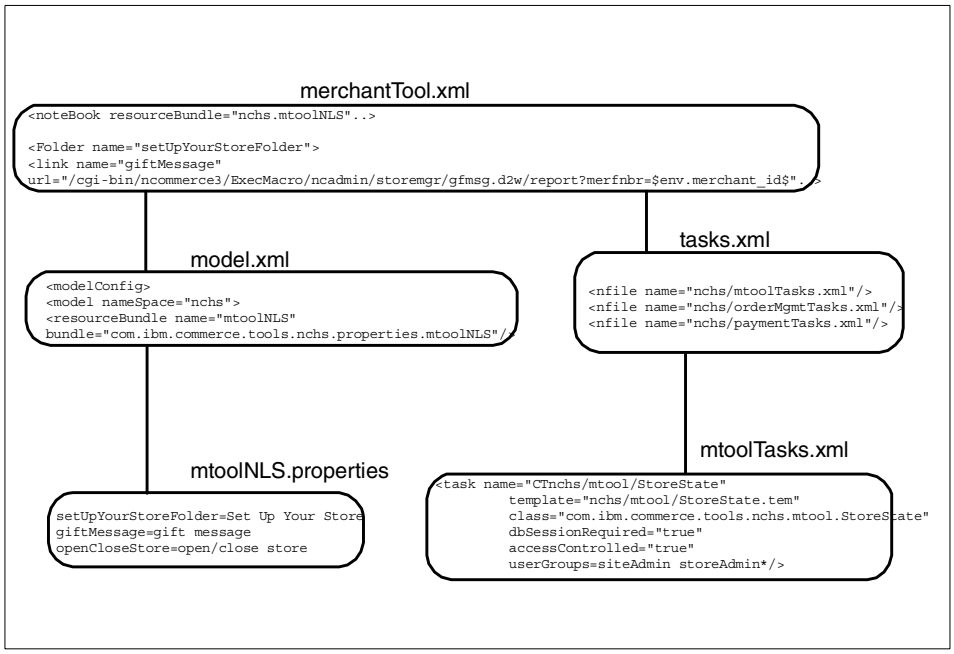


Figure 20. Relationship among xml files

Further information on XML

The following are further sources of information on XML:

- From your WCS application:
  `http://<your_hostname>:9527/doc/whatis/icxml4j.html`
- `http://www.ibm.com/developer/xml/`
- The redbook, *The XML Files: Using XML and XSL with IBM WebSphere 3.0*, SG24-5479

3.1.1.6 Multi Purpose code Generation language (MPG)

Note

The use of MPG is not supported by IBM. Any modification to files in relation to MPG is at your own risk. Changes made to files in relation to MPG may not work in previous and/or new versions of WebSphere Commerce Suite, Service Provider Edition. IBM does not guarantee any migration path for changes made to files in relation to MPG.

MPG (Multi Purpose Generator) is a utility used to generate output, and, in the case of WCS, it produces the pages in the merchant tool, payment wizard, and store creation wizard. It consists of two components: The model (Java class) and the template. The model consists of the data or variables that are used to generate the output, and the template describes the presentation of that output. Unfortunately, the MPG models in the WCS SPE Java framework are unpublished and cannot be modified nor can new ones be created. However, the template part of MPG consists of text files that are relatively simple to program; so, customization of the graphical layout is possible. Even though MPG is used throughout the merchant tool and wizards, new pages can be added in whatever Web technology the ISP is familiar with and supported by WCS, such as simple HTMLs, net.data macros, servlets, or JSP pages. MPG is also described in Appendix A, “Multi purpose code generation language” on page 243.

3.1.1.7 Net.Commerce architecture

The WCS SPE is built upon and extends the IBM award winning merchant server software, Net.Commerce 3.2. To understand WCS SPE, it is also important to understand the underlying Net.Commerce capabilities. Net.Commerce is built in two layers as shown in Figure 21.

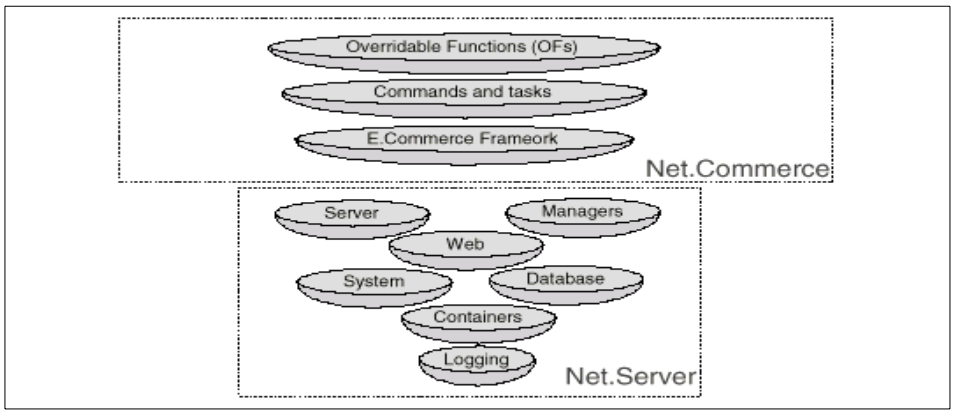


Figure 21. NetCommerce Architecture

Net.Server is a generic infrastructure containing foundation classes for the system. It forms an environment for components to plug into to allow it to handle Web requests and RDBMS-based transactions, among other things. The second layer is the Net.Commerce layer, which consists of a commerce-specific object model and a collection of commerce-related commands, tasks, and overridable functions (OFs). It is at this level that

possibilities for merchant customizations appear. Figure 22 describes how commands use tasks to call OFs.

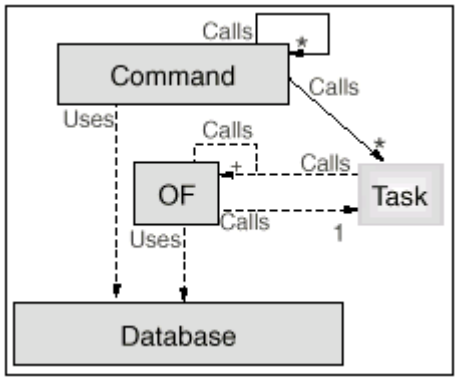


Figure 22. Commands, Tasks and Overridable Functions

Through the e-commerce framework, these OFs can be customized and replaced by merchant-specific code, and new commands and tasks can be created. This level of customization is used to implement business logic, such as special prices, once a customer buys 10 of a certain product. Typical skills required are C++, SQL, and knowledge of the Net.Commerce e-commerce framework.

Further information can be obtained from the following sources:

- The downloadable document, *Commands, Tasks, Overridable Functions and the e-commerce Data Objects* Updated 13 September, 1999 from the following Websites:
  `http://www-4.ibm.com/software/webservers/commerce/servers/lit-tech-general.html` or
  `http://www.software.ibm.com/commerce/net.commerce`
- The Redbook *Building e-commerce Solutions with Net.Commerce: A Project Guidebook*, SG24-5417.
- The book, *e-business with Net.Commerce* by Samantha Shurety, Prentice Hall. ISBN 0-13-083808-x

Chapter 4. Building custom stores in a shared environment

WCS SPE is based on the Net.Commerce product, which gives the ISP and merchants many possibilities to customize their site. Most of the customization that the merchant can do is done using the merchant tool. However, adding features to WCS SPE or offering further customizations will add a great value for the ISP because this will make one ISP different from another.

The ISP might want to add new features or offer specialized customizations for several reasons. One important reason is that adding your own new features is close to producing your own product, which we will refer to as branding WCS SPE.

Using WCS SPE as basis for a hosting environment, you will have to keep in mind that all merchants will run in a shared environment. This means that all merchants are using the same database, the same Web server, and also sharing a lot of code implementing the business logic. Enabling or using certain customizations might then have an impact on the whole environment, which you need to be aware of before offering more advanced customizations to your customers. When adding new features to the product, you also have to consider if the new features should be included in the merchant tool, have their own admin application, or just be offered as a service.

Also, when offering customizations to your merchants, you should consider if it is of general interest, in which case the customizations should be available in the merchant tool or, if it is so special, for one particular merchant.

In this chapter, we will explain the different levels of customizations that can be done with the product and what influence they will have regarding the other merchants and enabling certain kinds of customization. Since WCS SPE is a complex product, extensible knowledge is required to fully utilize all the features that come with the product, which means that a good understanding of HTML, Java Script, Net.Data, Java, and C++ is a must.

4.1 The merchant store model

The stores created in WCS SPE are running in a hosted environment, which means that they will share many common elements. The merchant tool and the store model are shared by all the stores and are the elements you would most like to customize. To get an understanding of how the stores makes use of the shared elements, we will start with a description of what happens when a new store is created using the wizard.

When a merchant creates a new store, one of the first things that happens is that a new record in the table merchant is created, and the merchant is assigned a unique reference number. This reference number is used to create a directory that will contain the HTML and other data for the merchant.

In this example, we just created a store using the basic store creator and gave it the name, advancedstore. The ID that will be used to create the directory can be found in the MERFNBR column in the MERCHANT table.

As Figure 23 shows, our newly-created advanced store got the reference ID, 1098.

```
To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to itso

   Database Connection Information

 Database server        = DB2/6000 6.1.0
 SQL authorization ID   = DB2INST1
 Local database alias   = ITSO

db2 => select merfnbr from merchant where mestname='advancedstore'

MERFNBR
-----------
       1098

  1 record(s) selected.

db2 =>
```

Figure 23. Fetching the merchant ID from the database

The directory that is created for the store is /usr/lpp/NetCommerce3/CHS/source/1098, which, by default, contains a file, settings.xml, that contains some default values for our site and the information that was entered using the store creator wizard.

The created directory also contains another directory, HTML, which, by default, contains a few HTML files that the store creator wizard copied. This HTML directory is also the directory that we can access from the merchant tool when managing files.

All the pre loaded files that you can see in Figure 24 on page 53 are just simple redirections to some commands with the proper parameters; so, if a user tries to access our store using the URL, `http://HOSTNAME/advancedstore/catalog`, the file, catalog.html, will redirect the user to the `CatalogDisplay` command. The pre loaded files will not be overwritten by WCS; so, the merchant can modify or delete them if they want to. See Figure 24.

```
f50>:/usr/lpp/NetCommerce3/CHS/source/1098/html>ls -l
total 56
-rw-rw-r--   1 nobody   nobody       605 Mar 09 13:42 catalog.html
-rw-rw-r--   1 nobody   nobody       604 Mar 09 13:42 customer_service.html
-rw-rw-r--   1 nobody   nobody       661 Mar 09 13:41 logon.html
-rw-rw-r--   1 nobody   nobody       600 Mar 09 13:41 register.html
-rw-rw-r--   1 nobody   nobody       610 Mar 09 13:42 search.html
-rw-rw-r--   1 nobody   nobody       603 Mar 09 13:42 shop_cart.html
-rw-rw-r--   1 nobody   nobody       610 Mar 09 13:42 zzz.html
f50>/usr/lpp/NetCommerce3/CHS/source/1098/html>
```

Figure 24. The preloaded HTML files

If the merchant has created new HTML or image files for the store, these can be uploaded to the same directory where the preloaded files exist (see Figure 24) by using the upload facility in the merchant tool. You have to keep in mind that managing files with the merchant tool is only possible for advanced stores. If you want merchants using the basic store to be able to upload new files, you will have to set up some kind of ftp service.

Using the merchant reference number, the store creator wizards also makes some other directories to keep store-specific macro files. The directories in our example, with reference number 1098, are as follows:

```
/usr/lpp/NetCommerce3/macro/en_US/1098/
/usr/lpp/NetCommerce3/macro/en_US/category/1098/
/usr/lpp/NetCommerce3/macro/en_US/product/1098/
```

Based on the information given during the create store process, a set of default macros will be placed in all the macro directories. As with the preloaded HTML files, the preloaded macro files can be overridden if necessary.

If you look at the preloaded macro files, for example, the logon.d2w macro, you will see something like this:

```
%include "1098/include.inc"
%include "1098/theme.inc"
%include "ClassicStoreModel/logon.d2w"
```

The files, include.inc and theme.inc, are both located in the same directory as the logon.d2w file and contain information about this particular store based on information given when the merchant created the store or if they later changed the store design using the merchant tool. This means that you should never edit these files yourself because your changes would be overwritten by the merchant tool.

The interesting part is the last include statement, which includes the actual content of the logon page based on which store model the merchant has selected. What this means is that all stores using the same store model share the same macros. Making changes in one of the default macros for a store model will be reflected immediately in all the stores using that model.

Apart from the merchant reference number in the table MERCHANT, a number of other database entries is also created. The LogonID and password entered on the first page in the store creator wizards are added to the SHOPPER table together with other administrative info in the SHADDR table, and an entry is made in the MCSPINFO table for the store.

4.2 Customizing the look of a store

The first thing a merchant probably would try to do is change the visual appearance of the shop. If the merchant created a store using the advanced store creator, a lot of customization can be done at the store level using the merchant tool.

After the store is created, the merchant will still be able to change the look of his or her store by using the design store feature in the merchant tool. This features lets the merchant change, create, upload, and manage files.

Since each store has its own directory for HTML pages, based either on the internal merchant reference or the store name, changes made to each store does not have any side effect on the other stores.

Right now the shopping flow is predefined and cannot be changed by the merchant, but, in some cases, the merchant wants a completely different shopping flow. This could, for example, be for easier navigation, quicker shopping, or something different.

Since all the stores are running in a shared environment, they all share the same macro files based on what store model they have chosen during initial store creation. However, there are some ways to do this kind of customization, even if the stores are running in a shared environment. One

possible solution is to add the feature as a new store model by customizing the wizard and making it available to all your customers.

It is also possible to customize at the store level, which is needed if, for example, a merchant only wants some small changes that should not be available to all other merchants. When the store was created, a few directories for merchant-specific macro files were created. These directories also contain some preloaded macros that contain a few lines of code to include merchant-specific settings together with the default store model macro. To customize one of the default macros for a merchant, such as the logon page for merchant 1098, perform the following steps:

1. Locate the default macro in either `/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel` or `/usr/lpp/NetCommerce3/macro/common/ExpressStoreModel` depending on which model the merchant uses.
2. Copy the macro file, logon.d2w, to a temporary working directory.
3. Rename the macro file to logon.inc
4. Edit the macro making the required changes.
5. When finished, copy the new macro, logon.inc, to the merchant-specific directory, /usr/lpp/NetCommerce3/macro/en_US/1098.
6. In the same directory, open logon.d2w and change the last line to:

   ```
   %include "1098/logon.inc"
   ```

   and save the file. This line tells Net.Commerce to use our customized version of the logon page instead of the shared macro.

After the change, you can launch the browser to verify that it is only this particular store that has the customized version of our logon page.

As this example shows, it is possible to customize each store independently of the other stores. However, customizing the look and feel of a store this way means that the merchant will no longer be able to change the look of the store by using the merchant tool. In fact, the changed files might even get overwritten by the merchant tool.

4.3 Enabling Net.Commerce features

Enabling some of the Net.Commerce functionality can sometimes add extra value for a merchant. What does Net.Commerce offer, and will it have any impact on the other merchants, that this particular merchant wants to have a special Net.Commerce feature enabled?

Net.Commerce as a basic product includes support for features such as shopper groups, discount and other shipping algorithms. However, if you want to enable features that are not directly available in the merchant tool, you might end up having to customize it yourself as an ISP. This is the case with shopper groups and shipping product by weight instead of quantity, which we in later chapters will describe in details how to enable in the WCS SPE.

When customizing beyond what the merchant tools allows your merchant to do, you should consider using a staging server in combination with your production environment. As most online stores operate 24 hours a day, 365 days of the year, this makes it difficult to perform maintenance or test changes to the system, without taking the system offline for a period.

Using a Net.Commerce staging server allows you to copy the production server database so that it is possible to test updates without affecting customers. This is not only useful for testing updates to the product catalog, but it is also very important for testing new commands, overridable functions, and macros.

4.4 Adding new Net.Commerce functionality

Extending Net.Commerce by adding new functionality is the most advanced customization that can be done and will be required for very specific kinds of customization, such as linking to legacy systems, implementing new shipping algorithms, or other kinds of functionality that cannot be implemented using standard Net.Commerce.

Customizing beyond the default Net.Commerce requires some work to be done in terms of specification, design, and implementation. Since this type of customization concerns the core of Net.Commerce, it requires knowledge about the framework and how the core components can be customized.

4.4.1 Net.Commerce architecture

The core Net.Commerce system consists of the components, commands, tasks, and overridable functions, that, together, are the object model that the merchant can customize.

Commands

Commands perform specific kinds of business operations, such as putting items in your shopping basket, displaying a category page, or updating your address book. The command and its parameter are sent to the Web server, typically posted from a form or as a hyper link, and then executed as an HTTP

request. When the command finishes executing, it sends back an HTTP response to the browser.

Net.Commerce commands are classified as one of the following two types, depending on whether they perform a Display or Process function.

Basically, Display commands just retrieve information from the database and display the information to the shopper in the form of HTML pages. They work by setting a view task that calls an overridable function to display the page. Display commands can be invoked consecutively and will show the same page every time. No changes are made to the database when a display command is executed, or the same change is made every time the command is executed. Examples of display commands are: AddressForm, CategoryDisplay, ProductDisplay, and OrderDisplay.

Process commands process and write data to the database. This type of commands will typical set several process tasks, which will call an overridable function to perform part of the overall processing. Not all updates to the database is done in the overridable function, so a process command itself can access the database and make the updates. Unlike the Display commands, the Process commands are not repeatable, so each time a command is executed the database is updated and if the command executes successfully, the changes are committed to the database. Examples of process commands are: AddressAdd, RegisterNew, OrderItemUpdate, and OrderItemDelete.

Most commands in Net.Commerce take as input parameters that are used either by the command itself, an overridable function, or another command that is called later. Two types of parameters exists: Required and optional. For the commands supplied with WCS SPE, the syntax diagrams and descriptions can be found in the manual, *Commands, Tasks, Overridable Functions and Database Tables*, which comes with the product.

Tasks

A task is a “contract” between a command and an overridable function or between two overridable functions. It defines the rules that govern the relationship between the two. These rules dictate the following:

- The work that the caller expects the overridable function to perform
- The parameters that the caller passes to the overridable function
- The parameters and other results that the caller expects the overridable function to return

When you create overridable functions, you must follow the rules that are defined for the task that the overridable function implements.

As with commands, there are different types of tasks: Display tasks, process tasks, and error tasks. Display tasks can be divided further into view and exception tasks.

- View tasks are used for overridable functions that display pages that shoppers see during the shopping process. For example, product pages, category pages, and registration pages.
- Process tasks are used for overridable functions that process information, such as calculating the total cost of an order, calculating the shipping charge, or doing online payments.
- Exception tasks are used for overridable functions that handle exception conditions, such as when a shopper tries to order an item that is not in stock.

Tasks also provide the scope and security framework so that, if a merchant decides to customize an overridable function for a specific task, the customized version of the function is only active in this particular store. This means that every merchant can implement and assign his or her own overridable function for the tasks.

Overridable functions

An overridable function is program code that implements a task. It implements the behavior that is expected by the task by dealing with input and output parameters as defined by the task.

When an overridable function is called, a task name and merchant reference number are provided to the overridable function manager. If the task is defined with a scope of 0 (mall scope), the merchant reference number is ignored, and default implementation of the overridable function is invoked. If the scope is 1 (store scope) and the merchant has provided an implementation, the merchant's implementation is invoked.

Functions that implement view or exception tasks usually populate the HTTP response back to the browser by calling one of the supplied overridable functions.

Overridable functions for process tasks usually process information, such as checking address information, calculating a product price, or even linking to external legacy systems.

If the merchant wants to customize one of the supplied overridable functions, they should replace the default implementation with their own. Even though all process tasks can be replaced, Net.Commerce also supplies a couple of *extendable* process tasks. These tasks are available to the merchant for performing additional processing prior to the completion of a command. Examples of such functions are:

- EXT_SHIPTO_UPD
- EXT_ORD_PROC

These two tasks are used in the `OrderItemShipTo` and `OrderProcess` commands and can be customized. By default, they are assigned to the overridable function, DoNothingNoArgs. The customer can then replace these functions to meet their business needs, and they can do it without interfering with the other merchant in the same environment. A typical example of customizing the extendable process task is integrating Net.Commerce with a legacy system, such as an order processing or online payment system.

4.4.2 Using Overridable functions or commands

In most cases, when a merchant needs some kind of customization at the Net.Commerce level, one of the supplied overridable functions for a task can be modified and then replace the original function. However, in some cases, none of the existing process tasks can be used for customization, and the only solution might be to write a completely new command for the required business logic.

With the tasks model, the scope for overridable functions can be defined at a store or mall level, which makes it possible for each task that a command calls to have more than one implementation, depending on the merchant requirement. Working with commands does not offer a similar feature, which means that new commands are available to all merchants in a shared environment. This must be taken into consideration when determining whether to create a new command or an overridable function.

Deciding if the customization should be in the form of a new command or an overridable function depends on what the merchant wants to accomplish. In general, a command should be created for store operation or if the feature needed is not supported by an existing command. A new overridable function should be created if you want to change the way an existing command operates or if a merchant has some special needs for a given task.

However, this rule should only be taken as a general guideline, since the reverse might be true for some kind of business. Table 1 summarizes the main differences between a command and an overridable function.

Table 1. Command and overrridable functions compared

	Command	Overridable function
Purpose	Defines business logic, such as adding items to shopping basket.	Performs a defined unit of work, as part of a command, such as fetching the price of a product.
Invocation	Invoked by the shopper via a button, hypertext link or a form, such as add to basket button.	Invoked by a command or another overridable function, using the task model.
Implementation	At mall level, that means that a merchant can't add a new command only available at his or her store.	At merchant level, each merchant can implement their own custom function.

Chapter 5. Shopper groups

Some existing business very likely have some system to differentiate regular shoppers from one-time-only shoppers. One common way to do this is to supply regular shoppers with a club card that gives them a discount on certain products each time they shop. In addition, if they are part of the club, these shoppers may be presented with special offers or products to which the one-time-only shoppers do not have access.

In any e-business, there are many parallels to the preferred shopper group schema. Instead of regular and one-time-only shoppers, an e-business normally has registered and guest shoppers. Similar to the application for a club card, registered shoppers must supply information, such as their name, billing address, shipping address, and phone number. In return, they are given (or personally create) a unique login name and password to your site. This functions as their club card and places them in your virtual store's preferred group. Thus, by logging in each time they come to your store, they will have access to specials and discounts that guest shoppers will not be able to access.

Another purpose of club cards is to enable businesses to keep track of shoppers' buying patterns. By doing this, businesses can send promotional information on products that fit into a customer's particular pattern in hopes of generating more business. The underlying idea is to personalize the shopping experience for the customer by providing product information that is most pertinent to his or her buying patterns.

In your e-business, personalization can be accomplished in a much more efficient manner. Because your system is constantly monitoring the products customers place in their shopping carts, it can instantaneously offer related products or accessories for existing products. Personalization can also go beyond just the sale of products. Registered shoppers could also possibly change the look and feel of the virtual store to suit their preferences.

However, it is important that personalization does not become too intrusive. For example, shoppers could get very annoyed if a related product is pushed onto the screen every time they access their shopping cart. An alternative would be to offer just one or two products during the checkout process. As always, it is important to ensure that the customer has a good shopping experience.

In this chapter, we explain how the ASP can customize the WCS to take advantage of those features of Net.Commerce.

5.1 Customizing WCS to offer shopper group features

By default, the WCS does not provide any tool for the ASP to offer the option of the shopper groups.

Don't forget that the basement of the WCS is the Net.Commerce 3.2; so, we will try to use and add the shopper group feature to it.

In Net.Commerce 3.2, being part of a shopper group can entitle shoppers to discounts or other bonuses for purchasing products. For example, if market research has shown that certain shoppers repeatedly purchase certain products, you could assign these shoppers to the same shopper group, or you can create a shopper group to reward frequent shoppers for their business. You could also create shopper groups that are based on demographic characteristics, such as a shopper group for seniors.

In Net.Commerce 3.2, you assign different prices to products for different groups of shoppers. You can also customize the way products and categories appear to members of shopper groups by creating a separate template with the Template Designer. The customized product template is assigned to the shopper group on the Product/Item Information form.

In Net.Commerce 3.2, you can create, search for, list, modify, or delete shopper groups using the Shopper Groups form.

Note

In Net.Commerce 3.2, the shoppers can belong to only one group per store.

To enroll a shopper in a shopper group after it is created, you must select the group from the Customer Information form.

To make it possible to offer this feature to the merchants, the ASP must make several customizations, additions, and changes to some parts of the WCS. We will work with tables, macros, and commands; we recommend that you follow the steps very carefully because they are common to all the stores in the mall.

5.1.1 Shopper group basic

First of all, we must remember that, in WCS, we will have several people involved in the shopping process.

- The *administrator* is a person who works for the ASP and has the responsibility of managing and controlling all of the WCS mall.
- The *merchant* is a person or enterprise that buys a store (a space in the WCS mall) to sell goods or services on the Internet. The Merchant can only have control over his or her own store.
- The *customer* is a person who gets into the merchant's store to buy goods and services. The customer can be registered on the Merchant's store or not, but, if the Customer has been registered in the store, he or she can belong to a shopper group (with other customers). The customer can only have control over his or her own orders.

To make this easy to understand, imagine that we are an enterprise that wants to offer some products over the World Wide Web (WWW). We contract the services of an ASP who hosts our store.

We design our store, and, finally, the store is working and receiving orders from several customers. Some of them are Registered Customers and the rest are not.

We note that some of these Registered Customers have several things in common: They belong to a sport club; so, we decide to contact them and offer membership in a Shopper Group so that they can earn better prices and special offers and discounts.

This is how we can do this: Every time a user who belongs to the Shopper Group logs onto the store and browses the catalog, the store automatically presents the special prices.

As we mentioned at the beginning of this chapter, Net.Commerce 3.2 (the basement of the WCS) has this feature, but, in this case, we must customize it.

Luckily, we can to use several code pieces , macros and commands from Net.Commerce 3.2 without problem.

As we see in Chapter 2.2, "Shop directory structure" on page 39, there are some files which are common for every store in the mall. Those files are located as shown in Figure 25 and Figure 26 on page 64.

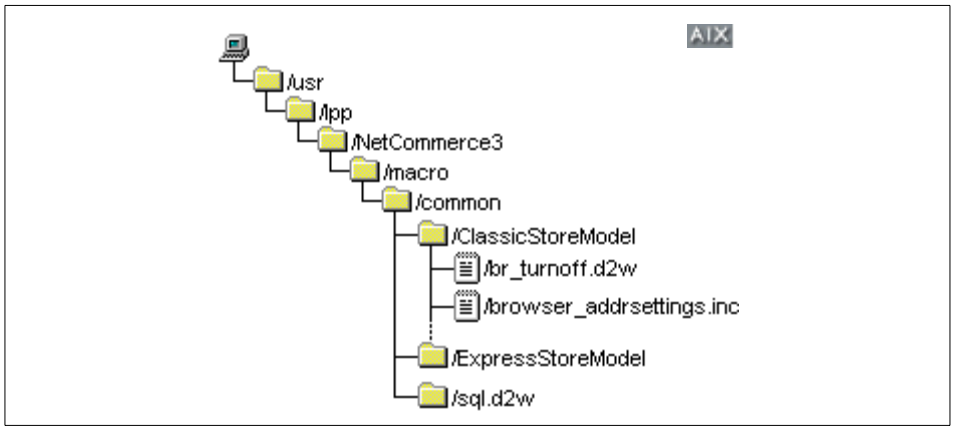


Figure 25. The common files and directories on AIX

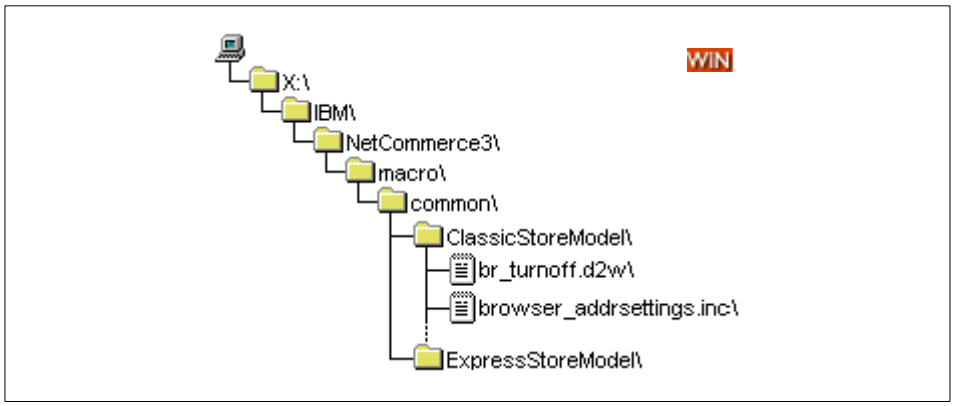


Figure 26. The common files and directories on WINDOWS NT 4.0

Note:

This customization can be made only in the Classic Store Model stores, because the Express Store Model the customers never been registered.

Another thing to remember is that most of the tables from the database are the same as for Net.Commerce, and they are common for every store in the mall. We can, therefore, use them for this customization.

The tables we need to this customization are:

- The Category table, CATEGORY, contains information that describes the product categories and subcategories for each store. Each row describes a category. The columns, CGLDESC and CGDISPLAY, are available for merchant customization. The content, path, and name of the template that displays the category information, subcategory information, and images that were originally stored in these fields are now stored in the table, CATESGP.
- The Shopper Group Category Template table, CATESGP, contains information indicating that the Net.Commerce system needs to display category information to shoppers in different shopper groups. Each row associates one shopper group with one template for one category. A category template defines the format in which categories are displayed. Template customization by category was introduced in Version 2 of Net.Commerce. It functions similarly to the existing shopper group product template customization, which uses the table, PRODSGP.
- The Category Product Relationship table, CGPRREL, defines the relationships between the categories and products. Each row describes one relationship.
- The Category Relationship Table, CGRYREL, defines the relationships between the categories and subcategories for each store. This information is used to structure the product categories that are presented to shoppers. Each row describes one relationship.
- The Merchant Customer Information table, MCUSTINFO, has two purposes:

  a. It associates shoppers with shopper groups.

  b. It holds additional information about shoppers who have special relationships with the merchant. For example, if a particular shopper has a customer number with the merchant, the number can be identified in this table.

  But remember: Each shopper and customer can belong to a maximum of one shopper group per merchant.
- The Merchant Profile table, MERCHANT, describes each merchant including information on the primary contact for the merchant. Each row corresponds to one merchant. For a one-merchant mall, this table contains only one row.
- The Shopper Group Product Template table, PRODSGP, contains information indicating that the Net.Commerce system needs to display product and item information to shoppers in different shopper groups. Each row associates one shopper group with one template for one

product. A product template defines the format in which products and items are displayed.

- The Product table, PRODUCT, describes all the products and items available at all stores. An item is a product that must be qualified by one or more attributes to be resolved into an SKU. An SKU (Stock Keeping Unit) is an orderable item. In this table, items are distinguished by having a non-null PRPRFNBR. Each row contains information on one product or item.
- The Shopper Address Book table, SHADDR, serves as an address book for each registered shopper. At the time of registration, the shopper provides his or her address, and this entry is flagged as permanent. When a shopper moves, the shopper can provide a new address, and a new entry is added to the table. The old address is not discarded, but it is flagged as temporary. Temporary rows are also created if a shopper provides a new address for a specific order without updating the address book.
  Shoppers can also add addresses for other individuals, such as relatives or places, to this table. All such entries are flagged as permanent.
- The Shopper Group table, SHOPGRP, describes the shopper groups that are defined for each store. Each row contains information on one shopper group.
- The Shopper Profile table, SHOPPER, contains information needed to identify each shopper and user to the Net.Commerce system. It also contains some basic contact and classification information. (More contact and classification information can be found in the SHOPDEM and SHADDR tables.)
  When the database is initially installed, a row is inserted into the SHOPPER, ACCTRL, SHOPDEM, and SHADDR tables defining the site administrator. In the table, SHLOGID is set to ncadmin, and SHLPSWD is set to the encrypted form of ncadmin. The administrator should change this password immediately after installing the system.
  One row is defined for each shopper (whether or not the shopper is registered) and for each user.

The relationship between the category, product, and item tables is shown in Figure 27 on page 67.

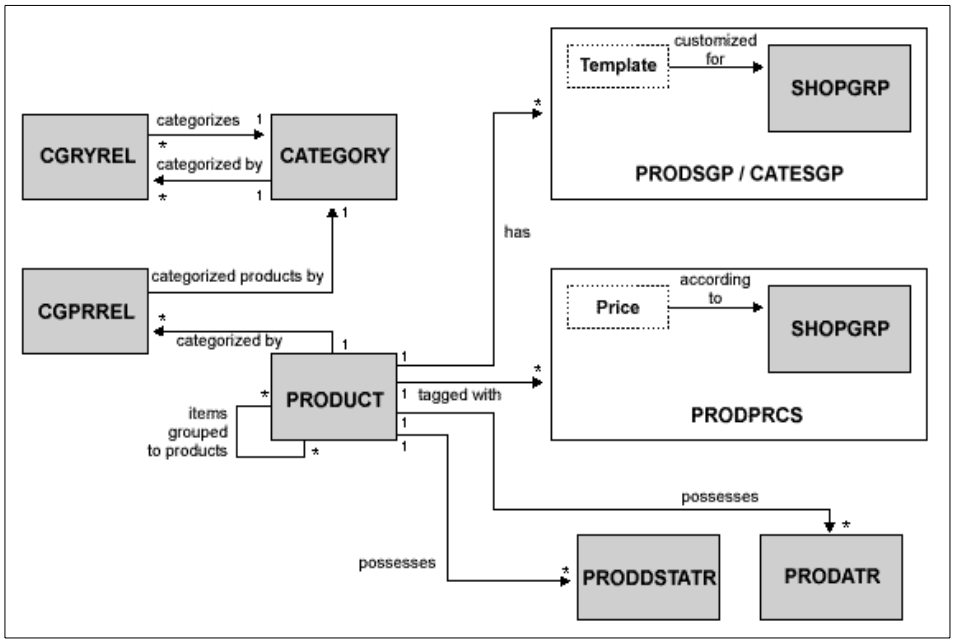


Figure 27. Data model for categories, products and items

The relationship between the merchant, store, shopper group, and customer tables is shown in Figure 28.

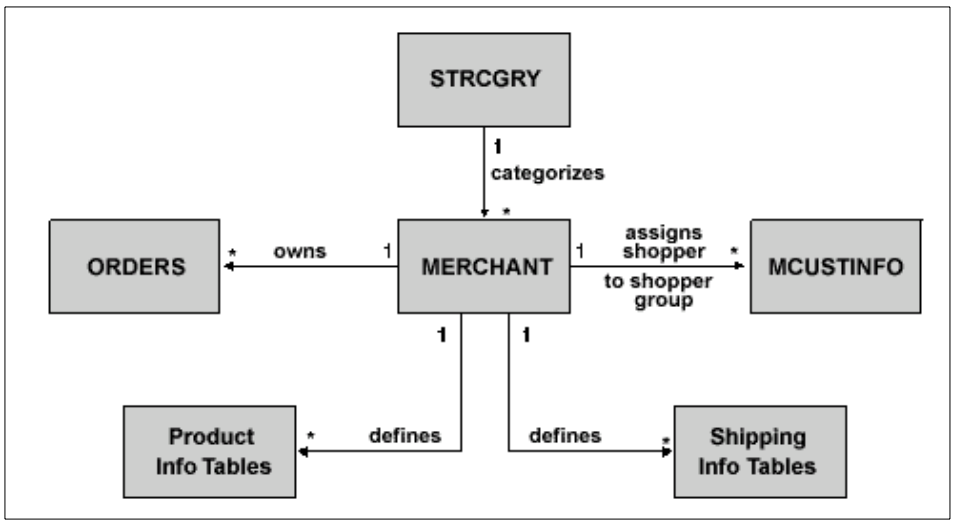


Figure 28. Data model on merchants, stores, shopper groups, and customers

Every time a *Registered Shopper* wants to get in to the store, the servlet Logon (managed by the Websphere Application Server -WAS-) invokes the Net.Commerce command, `LogonForm`.
The `LogonForm` command displays a page that allows a registered shopper to log on to the store or mall.
Once the Registered Shopper types the username and password, the WCS proceeds to execute the `Logon` command.
The `Logon` command logs a registered shopper onto a store or mall; so, the WCS shows the catalog using the `CategoryDisplay` command. This command displays a category page based on the shopper group in which the shopper is a member and verifies which template must be used.
This command is called the `cat_category.d2w` macro and is located on the path, X:\IBM\NetCommerce3\macro\en_US\category\<number of the store>, (for Windows NT 4.0) or
usr/lpp/NetCommerce3/macro/en_US/category/<number of the store> (for AIX).
The macro consists of three Net.Data code lines, and appears as shown in Figure 29.

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_category.d2w"
```

Figure 29. The macro cat_category.d2w after the changes

In this figure, we can see that the first two lines call an environment macro for the store (in our example, <number of the store> has been replaced by the number 831). The cat_category.d2w macro (see the last line in Figure 29) is the macro that shows the categories, sub-categories, and the product list.
Once the Registered Shopper chooses a category and/or subsequent subcategories, the WCS shows the product list.
When the Registered Shopper choose a product, the `ProductDisplay` command is activated. The `ProductDisplay` command displays a product or item page based on the shopper group of which the shopper is a member (if any). As in the case of the *CategoryDisplay* command, this command calls for the cat_product.d2w macro, located on the path, X:\IBM\NetCommerce3\macro\en_US\product\<number of the store> (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/en_US/product/<number of the store> (for AIX). Here again, the macro consists of three Net.Data code lines, and it looks like that shown in Figure 30 on page 69.

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_product.d2w"
```

Figure 30. The macro cat_product.d2w after the changes

In this figure, we can see that the first two lines call an environment macro for the store (in our example, <number of the store> has been substituted by the number 831). The cat_product.d2w macro (see the last line in Figure 30) is the macro that shows the categories, sub-categories, and the product list. Finally, if the Registered Shopper chooses a product, the `OrderItemList` command displays a list of products and items in the shopping cart from which shoppers can select the ones they want to order. From this point, the process is the same for all the shoppers.

5.1.2 The customization

The WCS customization to offer the shopper group features can be split into two parts: The first part consists of the preparation of templates, macros, and everything necessary to prepare the *backoffice* (the ASP side), and the second part consists of the customization of the merchant tool to make it possible for the merchant to create, modify, and manage shopper groups and registered shoppers.

5.1.2.1 Customization of the site

The first step in this customization must be made by the ASP because the ASP is the only person with access to the macros.

Note

This first part of the customization is where the ASP prepare the way for the merchant to operate with the shopper groups features.

First, we will change the standard WCS registration. The default is to register at mall level so that a shopper is not connected to any store. For the shopper groups feature, we require that merchants can only access shoppers that are registered at their store or mall wide. Therefore, the standard registration must be changed. The MCUSTINFO table in WCS represents the relationship between the shopper and the merchant. See Figure 31 on page 70.

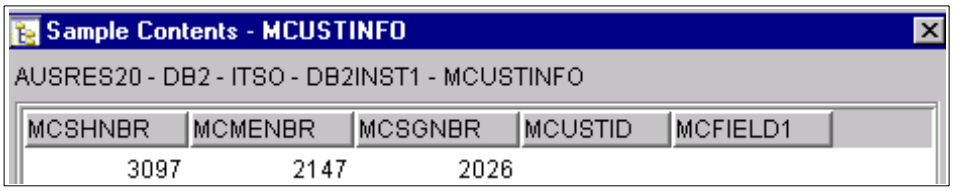


Figure 31. The MCUSTINFO table

Therefore, we require a command that will call the original RegisterNew command but will also create this row in table MCUSTINFO. We will also use the mcfield1 field as a flag, “A”, to signify whether a shopper is registered mall wide. You will see an example SQL in Figure 32 where user `shopper` is connected to merchant `bellavista`, but it is also registered mall wide (mcfield1 = “A”).

```
-------------------------- Command entered ----------------------------
select shlogid, mename, mcmenbr, mcshnbr, mcfield1
from shopper, merchant, mcustinfo
where shlogid='shopper'
and mcshnbr=shrfnbr
and mcmenbr=merfnbr
------------------------------------------------------------------------

SHLOGID    MENAME     MCMENBR     MCSHNBR     MCFIELD1
---------- ---------- ----------- ----------- --------
shopper    bellavista        2147        8068 A

  1 record(s) selected.
```

Figure 32. SQL for example mcustinfo row

Our new command is called `regWrapper`. You can find the full code and implementation details in Appendix B.1, “RegWrapper source code” on page 267. First, it calls the original `RegisterNew` command using the CommandManager as follows:

```
// get id for RegisterNew cmd
NC_RegistrationID ID_regnew("IBM","NC","RegisterNew",1.0);
// Use CommandManager to call RegisterNew cmd
if( NC_CommandManager::Call(LocalReq,LocalRes,Env,ID_regnew)!=NULL ) {
return false;
}
```

Then, it creates a new row in the MCUSTINFO table using the merchantRefNum parameter, the shopper’s reference number, and an empty shopper group value.

```
String _VAL_shopref = loggeduser->getRefNum();
String _VAL_shopgrp = "";
mcustinfo = mcustinfo_home.Create(merchantRefNum, _VAL_shopref
,_VAL_shopgrp);
```

It also updates mcfield1 to the value *A* if the parameter regtype is equal to *1*:

```
if (_VAL_regtype == "1" )
mcustinfo->setField1("A");
```

Lastly, we need to change the registration macro for the store to implement our new command. In the ClassicStoreModel, this is file:

```
/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/reg_new.d2w
```

We change the call from RegisterNew to our new command, `regWrapper` (shown in bold below):

```
<FORM name="registerNew" method=POST
action="https://$(HOST_NAME)/cgi-bin/ncommerce3/regWrapper">
<INPUT TYPE="hidden" NAME="merchant_rn" Value="$(MerchantRefNum)">
<INPUT TYPE="hidden" NAME="old_email" Value="">
```

Then, we add our new parameter, regtype, as a check box for the shopper to click on (shown in bold below):

```
<TD ALIGN="left">
<FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_REGISTERNEW)</FONT>
</TD></TR>
<TR><TD></TD></TR>
<tr><TD COLSPAN=3>
<INPUT TYPE="checkbox" NAME="regtype" value='1'><FONT
FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">Check here if you want Mall
wide registration.</FONT>
</TD></tr>
<TR><TD></TD></TR>
```

The changed registration screen is shown in Figure 33 on page 72.

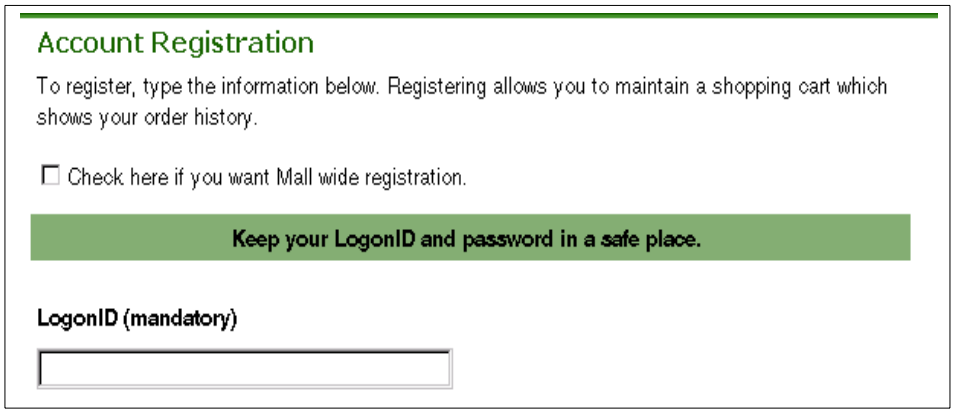


Figure 33. Store and mall wide registration screen

Note

The command regwrapper is only called when the store is in open state, if the user is in the merchant tool and looking at the store, regwrapper will not be called and the change will not be noticed.

Now, we continue with the required macro changes. To prevent any undesirable changes on the other stores that do not need or do not want the shopper groups featured, we must perform several steps:

When a store is created, the WCS give a consecutive number for this store; so, in the hard drive, we can see on locations X:\IBM\NetCommerce3\macro\en_US\category (for Windows NT 4.0) and usr/lpp/NetCommerce3/macro/en_US/category (for AIX) a subdirectory named with the number of the store (for example, our store has the number 831).

On this sub-directory, we can find the macro, cat_category.d2w, which contains a few lines of code as shown in Figure 34.

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_category.d2w"
```

Figure 34. The macro cat_category.d2w

On the first two lines, we can see that the macro includes another two environment macros, include.inc and theme.inc, which are located on the path X:\IBM\NetCommerce3\macro\en_US\ (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/en_US/ (for AIX) on a subdirectory with the same number as the store (for example, our store has the number 831).

Those macros declare several environment variables, such as background color, text color, buttons, and so on.

The last line is the most important because it invokes the macro that really does the work.

This macro is located on the path X:\IBM\NetCommerce3\macro\common\ClassicStoreModel (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/common/ClassicStoreModel (for AIX)

This is a common macro. It means that if you make some changes on it, these changes can affect the behavior of the other stores.

To prevent this, we can make a copy of this macro with a name, such as cat_cat<number of the store>.d2w (even in the same subdirectory) and make all the changes on it (for example, cat_cat831.d2w).

The only change we must make on the macro cat_category.d2w is to change the name of the macro in the last line,
%include "ClassicStoreModel\cat_category.d2w", for
%include "ClassicStoreModel\cat_cat831.d2w".

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_cat831.d2w"
```

Figure 35. The macro cat_category.d2w after the changes

Now, we can modify the macro, cat_cat<number of the store>.d2w, located in X:\IBM\NetCommerce3\macro\common\ClassicStoreModel (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/common/ClassicStoreModel (for AIX).

This macro is the template that presents the category information for all the stores.

By default, if you have more than one price for the same product, when the table of products is displayed, it shows one different row for each product; so, if you have two different prices (one normal price and a price for one shopper

group), in the browser, there will appear a double product list. One with the normal prices and the other with the prices for the shopper group.

If you see the report section of this macro, you can see that the macro has several functions to present the category information on the browser.

These functions are:

```
%function(dtw_odbc) GET_CATEGORY_TITLEINFO()
%function(dtw_odbc) DISPLAY_CATEGORIES()
%function(dtw_odbc) DISPLAY_PRODUCT_LIST_DUALPRICE()
%function(dtw_odbc) DISPLAY_PRODUCT_LIST_SINGLEPRICE()
```

In Figure 36 on page 75, we can see what those functions in the Report section are called.

```
%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{
.
.
.
<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth)
ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <TR>
        <TD ALIGN=$(TitleAlignment) VALIGN="center">
        %IF (PAGE == "LOGON")
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
            SIZE=$(TitleFontSize)><B>Welcome back $(SESSION_ID)!</B></FONT>
        %ELSE
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(TXT_TITLE_CATALOG)</B></FONT>
        %ENDIF
        </TD>
    </TR>
</TABLE>
</CENTER>

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment)
CELLPADDING=4 CELLSPACING=0>

    <TR><TD>@DISPLAY_CATEGORIES()</TD></TR>

    %IF (ConvMultiplyFactor == NULL || ConvDivideFactor == NULL)
        <TR><TD>@DISPLAY_PRODUCT_LIST_SINGLEPRICE()</TD></TR>
    %ELSE
        <TR><TD>@DISPLAY_PRODUCT_LIST_DUALPRICE()</TD></TR>
    %ENDIF

    %IF (NOCATEGORIESINCATEGORY == "YES"  &&  NOPRODUCTSINCATEGORY == "YES")
        <TR><TD><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(MSG_NOPRODUCTS)</FONT></TD></TR>
    %ENDIF

    <!--TR><TD ALIGN="center">
    <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)">
    <BR>Use our secure online ordering system, or call ...</FONT></TD></TR-->

</TABLE>
</CENTER>

</BODY>
</HTML>

%}
```

Figure 36. The Report Section of the cat_cat<number of the store>.d2w macro

We will modify the *DISPLAY_PRODUCT_LIST_SINGLEPRICE* function because this is the function that displays the product list and the prices for each product in the list in this case. As shown in Section 5.1.1, “Shopper group basic” on page 62, the database tables involved in this process are closely related; the function, *DISPLAY_PRODUCT_LIST_SINGLEPRICE*, must access up to three tables and several columns at the same time just to display the prices. Figure 37 shows the tables involved by this macro, product, cgprrel, and prodprcs (previously described), and, after a short analysis, we can see that the function never uses the table, prodsgp, which has the relation between the shopper group and this price.

```
%function(dtw_odbc) DISPLAY_PRODUCT_LIST_SINGLEPRICE() {

SELECT prfull, prnbr, prsdesc, prthmb, prldesc1, pppre, prrfnbr,
          cpseqnbr, cpcgnbr,
          ppprc,
          NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder),
$(CurGroupingSize) , $(CurDecimalPlaces), '', 1 ) as FormattedCurPrice
FROM   product, cgprrel, prodprcs
WHERE  cpcgnbr=$(cgrfnbr) and cpmenbr=$(cgmenbr) and cpprnbr=prrfnbr and prpub=1
          and ppprnbr=prrfnbr and ppmenbr=$(MerchantRefNum) and
          (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or
          (ppdeffs is null and ppdefff is null)) ORDER BY cpseqnbr ASC, pppre DESC
   %REPORT{
       <CENTER>
       <TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment)
       CELLPADDING=4 CELLSPACING=0>
       %IF (cgrfnbr != HomeCategoryNum)
       <TR>
          <TD align="left">
          <FONT FACE="$(BodyFontFace)"
          SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRYLIST)</FONT>
          </TD>
       </TR>
       %ENDIF
       <TR><TD><BR></TD></TR>
       %ROW{
          @DTW_CONCAT(V_FormattedCurPrice, $(CurDescription), SHOWPRICE)
          %INCLUDE "/ClassicStoreModel/cat_productlist.inc"
       %}
   </TABLE>
   </CENTER>
   %}
  %MESSAGE{
    100:{ @DTW_ASSIGN(NOPRODUCTSINCATEGORY, "YES") %} :continue
       default:{<BR>Problem with DISPLAY_PRODUCT_LIST_SINGLEPRICE()
       in file $(DTW_MACRO_FILENAME) because of error $(RETURN_CODE).
       <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
  %}
%}
```

Figure 37. Part of the function DISPLAY_PRODUCT_LIST_SINGLEPRICE()

If we add some lines to the function in the SQL statement that selects the variables from those databases, we can solve this part of the problem. See Figure 38.

```
%function(dtw_odbc) DISPLAY_PRODUCT_LIST_SINGLEPRICE() {

SELECT prfull, prnbr, prsdesc, prthmb, prldesc1, pppre, prrfnbr,
       cpseqnbr, cpmenbr, cpcgnbr, cpprnbr,
       pppre, ppsgnbr, ppprnbr, ppmenbr,
       psprnbr, pssgnbr,
       NC_CurrFormat(pppre, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), '', 1 )
as FormattedCurPrice
FROM   product, cgprrel, prodprcs, prodsgp
WHERE  CGPRREL.cpcgnbr=$(cgrfnbr)
        and CGPRREL.cpmenbr=$(cgmenbr)
        and CGPRREL.cpprnbr=PRODUCT.prrfnbr
        and prpub=1
        and (($(NC_NOW) >= PRODPRCS.ppdeffs and $(NC_NOW) <= PRODPRCS.ppdefff) or
       (PRODPRCS.ppdeffs is null and PRODPRCS.ppdefff is null))
        and PRODSGP.psprnbr=PRODUCT.prrfnbr
        and PRODPRCS.ppprnbr=PRODUCT.prrfnbr
     and PRODPRCS.ppsgnbr=$(OHT_CODE)
     and PRODPRCS.ppmenbr=$(MerchantRefNum)

ORDER BY cpseqnbr ASC, pppre DESC
.
.
.
```

Figure 38. The SQL statement of DISPLAY_PRODUCT_LIST_SINGLEPRICE()

We added the table, prodsgp, and some *where* conditions, such as `and PRODPRCS.ppsgnbr=$(OHT_CODE)`.

The ppsgnbr column of the PRODPRCS table is the key to getting the correct price for a specific shopper group.

The *$(OHT_CODE)* variable is the result from the other two functions that we included. Those functions are: *G*ET_SHOPERGROUP and GET_CODEPRICE.

The GET_SHOPERGROUP function gets the shopper group code from the table SHADDR using the variable $(SESSION_ID), which has the username of the shopper. If the shopper is not a registered shopper, the variable, $(SESSION_ID), has a code generated by the WCS.

```
%function(dtw_odbc) GET_SHOPERGROUP() {
SELECT sanick, sashnbr
FROM   shaddr
WHERE  SHADDR.sanick='$(SESSION_ID)'
%REPORT{
    %ROW{
        @DTW_assign(OHT_USER, V_SASHNBR)
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_USER, NULL) %} :continue
        default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME)
        because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}
```

Figure 39. The GET_SHOPPERGROUP() function

You can see that, if a shopper group exists, the variable, $(OHT_USER), gets a value; otherwise, if the shopper group does not exist (because it is not a registered user), the variable, *$(*OHT_USER*)*, gets a NULL value.

The GET_CODEPRICE() function, shown in Figure 40, uses the variable, $(OHT_USER), to get the code of the price belonging to the shopper group.

```
%function(dtw_odbc) GET_CODEPRICE() {
SELECT mcshnbr, mcsgnbr
FROM   mcustinfo
WHERE  MCUSTINFO.mcshnbr=$(OHT_USER)
%REPORT{
    %ROW{
        @DTW_assign(OHT_CODE, V_MCSGNBR)
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_CODE, NULL) %} :continue
        default:{<BR>Problem with GET_CODEPRICE() in file $(DTW_MACRO_FILENAME)
        because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}
```

Figure 40. The GET_CODEPRICE() function

Now, using the MCUSTINFO table, which lists the relationships between the shoppers and the shopper groups, the variable, $(OHT_CODE), gets the code of the shopper group, and, with this value, we can get the correct product price for a specific shopper group.

Finally, we modify the HTML_REPORT section of the macro. See Figure 41.

```
%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{
<HTML>
@GET_CATEGORY_TITLEINFO()
<HEAD>
.
.
.
<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment)
CELLPADDING=4 CELLSPACING=0>
    <TR><TD>@DISPLAY_CATEGORIES()</TD></TR>
    %IF (ConvMultiplyFactor == NULL || ConvDivideFactor == NULL)
      @GET_SHOPERGROUP()
           %IF (OHT_USER == NULL)
           <TR><TD>@DISPLAY_PRODUCT_LIST_SINGLEPRICE_ISNULL()</TD></TR>
       %ELSE
           @GET_CODEPRICE()
               %IF (OHT_CODE == NULL)
               <TR><TD>@DISPLAY_PRODUCT_LIST_SINGLEPRICE_ISNULL()</TD></TR>
           %ELSE
               <TR><TD>@DISPLAY_PRODUCT_LIST_SINGLEPRICE()</TD></TR>
           %ENDIF
           <TR><TD>@DISPLAY_PRODUCT_LIST_SINGLEPRICE()</TD></TR>
       %ENDIF
    %ELSE
        <TR><TD>@DISPLAY_PRODUCT_LIST_DUALPRICE()</TD></TR>
    %ENDIF
    %IF (NOCATEGORIESINCATEGORY == "YES"  &&  NOPRODUCTSINCATEGORY == "YES")
        <TR><TD><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(MSG_NOPRODUCTS)</FONT></TD></TR>
    %ENDIF
    <!--TR><TD ALIGN="center"><FONT FACE="$(BodyFontFace)"
    SIZE="$(BodyFontSizeSmall)"><BR>Use our secure online ordering system, or call
...</FONT></TD></TR-->
</TABLE>
</CENTER>
</BODY>
</HTML>

%}
```

Figure 41. The new HTML_Report section

The next step is to make a new function that manages those cases when the shopper is not a registered user.

This new function is called from the macro in Figure 41 and its name is *DISPLAY_PRODUCT_LIST_SINGLEPRICE_ISNULL*.

This function is almost the same as the DISPLAY_PRODUCT_LIST_SINGLEPRICE function, but, as we can see in Figure 42, we compare the ppsgnbr column with the prrfnbr column of the PRODUCT table instead of the $(OHT_CODE) variable.

```
%function(dtw_odbc) DISPLAY_PRODUCT_LIST_SINGLEPRICE_ISNULL() {

SELECT prfull, prnbr, prsdesc, prthmb, prldesc1, pppre, prrfnbr,
       cpseqnbr, cpmenbr, cpcgnbr, cpprnbr,
       ppprc, ppsgnbr, ppprnbr, ppmenbr,
       pspmnbr, pssgnbr,
       NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), '', 1 )
as FormattedCurPrice
FROM   product, cgprrel, prodprcs, prodsgp
WHERE  CGPRREL.cpcgnbr=$(cgrfnbr)
        and CGPRREL.cpmenbr=$(cgmenbr)
        and CGPRREL.cpprnbr=PRODUCT.prrfnbr
        and prpub=1
        and (($(NC_NOW) >= PRODPRCS.ppdeffs and $(NC_NOW) <= PRODPRCS.ppdefff) or
       (PRODPRCS.ppdeffs is null and PRODPRCS.ppdefff is null))
        and PRODSGP.pspmnbr=PRODUCT.prrfnbr
        and PRODPRCS.ppprnbr=PRODUCT.prrfnbr
     and PRODPRCS.ppsgnbr is NULL
     and PRODPRCS.ppmenbr=$(MerchantRefNum)
ORDER BY cpseqnbr ASC, pppre DESC
    %REPORT{
       <CENTER>
       <TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment)
       CELLPADDING=4 CELLSPACING=0>
       %IF (cgrfnbr != HomeCategoryNum)
       <TR>
           <TD align="left">
               <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRYLIST)</FONT>
           </TD>
       </TR>
       %ENDIF
       <TR><TD><BR></TD></TR>
       %ROW{
           @DTW_CONCAT(V_FormattedCurPrice, $(CurDescription), SHOWPRICE)
           %INCLUDE "/ClassicStoreModel/cat_productlist.inc"
       %}
    </TABLE>
    </CENTER>
    %}
  %MESSAGE{
    100:{ @DTW_ASSIGN(NOPRODUCTSINCATEGORY, "YES") %} :continue
       default:{<BR>Problem with DISPLAY_PRODUCT_LIST_SINGLEPRICE()
       in file $(DTW_MACRO_FILENAME) because of error $(RETURN_CODE).
       <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
  %}
%}
```

Figure 42. The DISPLAY_PRODUCT_LIST_SINGLEPRICE_ISNULL function

Now, it is time to work with the cat_product.d2w macro.

This macro works very similarly to cat_category.d2w, but, in this case, we only work with products, not with categories.

As in the cat_category.d2w macro, when a store is created, the WCS gives a consecutive number for this store; so, in the hard drive, we can see, on location X:\IBM\NetCommerce3\macro\en_US\product (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/en_US/product (for AIX), a sub-directory named after the number of the store (for example, our store has the number 831).

On this sub-directory we can find the macro, cat_product.d2w, which contains a few lines of code as shown in Figure 43.

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_product.d2w"
```

Figure 43. The macro cat_category.d2w

On the first two lines, we can see that the macro includes another two environment macros: include.inc and theme.inc.

Both are located on the path X:\IBM\NetCommerce3\macro\en_US\ (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/en_US/ (for AIX) on a subdirectory with the same number of the store (for example, our store has the number 831).

Those macros declare several environment variables, such as the color of the background, text, buttons, and so on.

The last line is the most important because it invokes the macro that really does the work.

This macro is located on the path X:\IBM\NetCommerce3\macro\common\ClassicStoreModel (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/common/ClassicStoreModel (for AIX)

This is a common macro. It means that if you make some changes to it, these changes can affect the behavior of the other stores.

To prevent this, we can make a copy of this macro with a name, such as cat_pro<number of the store>.d2w (even in the same subdirectory) and make all the changes to it (for example, cat_pro831.d2w).

The only change we must make on the macro cat_product.d2w is to change the name of the macro in the last line:
%include "ClassicStoreModel\cat_product.d2w" for
%include "ClassicStoreModel\cat_pro831.d2w"

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_pro831.d2w"
```

Figure 44. The macro cat_product.d2w after the changes

Now, we can modify the macro, cat_cat<number, of the store>.d2w located in X:\IBM\NetCommerce3\macro\common\ClassicStoreModel (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/common/ClassicStoreModel (for AIX).

This macro is the template that presents the product information for all the stores.

By default, if you have more than one price for the same product, when the table of products is displayed, it shows one different row for each product; so, if you have two different prices (one normal price and a price for one shopper group), in the browser, there will appear a double product list. One with the normal prices and the other with the price for a shopper group.

If you look at the report section of this macro, you can see that the macro has several functions to present the category information on the browser.

Those functions are:

%function(dtw_odbc) GET_PRODUCTNAME()
%function(dtw_odbc) GET_DISTINCTATTRIBUTES()
%function(dtw_odbc) DISPLAY_PRODUCT_IMAGE()
%function(dtw_odbc) DISPLAY_PRODUCT_INFO
%function(dtw_odbc) DISPLAY_PRODATTR
%function(dtw_odbc) DISPLAY_PRODUCT_DUALPRICE
%function(dtw_odbc) DISPLAY_PRODUCT_SINGLEPRICE()
%function(dtw_odbc) DISPLAY_PRODUCT_DUALPRICE_RANGE()
%function(dtw_odbc) DISPLAY_PRODUCT_SINGLEPRICE_RANGE()
%function(dtw_odbc) DISPLAY_ITEMS_DROPDOWN()

In Figure 45 on page 83, we can see how some of the most important functions of the Report Section are called.

```
%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{
<HTML>
.
.
.
<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment)
CELLPADDING=4 CELLSPACING=0>
<!--A HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/InterestItemAdd?product_rn=
$(prrfnbr)&merchant_rn=$(MerchantRefNum)&WISHLISTSHOPPER_RN=$(SESSION_RN)
&url=http://$(HOST_NAME)/cgi-bin/ncommerce3/InterestItemDisplay">
<FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)">Add to my wishlist</FONT>
</A-->
@DISPLAY_PRODUCT_IMAGE()
@DISPLAY_PRODUCT_INFO()
%IF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor == NULL ||
ConvDivideFactor == NULL))
@DISPLAY_PRODUCT_SINGLEPRICE()
%ELIF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor != NULL &&
ConvDivideFactor != NULL))
    @DISPLAY_PRODUCT_DUALPRICE()
%ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor == NULL ||
ConvDivideFactor == NULL))
    @DISPLAY_PRODUCT_SINGLEPRICE_RANGE()
    @DISPLAY_ITEMS_DROPDOWN()
%ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor != NULL &&
ConvDivideFactor != NULL))
    @DISPLAY_PRODUCT_DUALPRICE_RANGE()
    @DISPLAY_ITEMS_DROPDOWN()
%ENDIF
<TR><TD ALIGN="center"><FONT FACE="$(BodyFontFace)" SIZE="-1"
COLOR=$(HighlightFontColor1)>**You can change or remove items from your shopping
cart at any time.</FONT></TD></TR>
.
.
.
%}
```

Figure 45. The Report Section of the cat_pro<number of the store>.d2w macro

The function we will modify is DISPLAY_PRODUCT_SINGLEPRICE() because this is the function that displays the price of the product chosen by the customer from the product list.

As shown in Section 5.1.1, “Shopper group basic” on page 62, the databases involved in this process have a very close relationship. As shown in Figure 46 on page 84, the DISPLAY_PRODUCT_SINGLEPRICE function uses two tables and several columns at the same time to display the prices.

```
%function(dtw_odbc) DISPLAY_PRODUCT_SINGLEPRICE(){

    SELECT prnbr, ppprc, ppcur, pppre, ppdeffs, ppdefff,
           NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces),
'en_US', 1 ) as FormattedCurPrice
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum)
     and prrfnbr=$(prrfnbr)
     and ppprnbr=$(prrfnbr)
     and (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or (ppdeffs is null and
    ppdefff is null))
    ORDER BY pppre DESC
    %REPORT{
    @DTW_ASSIGN(ACTUALPRICE, "0")
    <TR BGCOLOR=$(HighLightColor2)>
        <TD ALIGN="center" COLSPAN=2>
    %ROW{
        %IF (ROW_NUM == "1")
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
            <B>$(LBL_PRODUCTPRICE) : </B>
            </FONT>
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
            $(V_FormattedCurPrice) $(CurDescription)</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
            @DTW_ASSIGN(ACTUALPRICE, V_FormattedCurPrice)
        %ENDIF
        %IF (V_pppre == "0" && ACTUALPRICE < V_FormattedCurPrice)
            <FONT FACE="$(BodyFontFace)" SIZE="-1">
            (<STRIKE>$(V_FormattedCurPrice) $(CurDescription)</STRIKE>)</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
        %ENDIF
    %}
```

Figure 46. Part of the function DISPLAY_PRODUCT_SINGLEPRICE()

As we can see, the tables involved by this macro are product, and prodprcs, which were described earlier, and, after a short analysis, we can see that the function never uses the table, prodsgp, which has the relation between the shopper group and this respective price.

If we add some lines to the function in the SQL statement that selects the variables from those databases, we can solve this other part of the problem. See Figure 47.

```
%function(dtw_odbc) DISPLAY_PRODUCT_LIST_SINGLEPRICE() {

SELECT prfull, prnbr, prsdesc, prthmb, prldesc1, pppre, prrfnbr,
       cpseqnbr, cpmenbr, cpcgnbr, cpprnbr,
       ppprc, ppsgnbr, pppmnbr, ppmenbr,
       psprnbr, pssgnbr,
       NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), '', 1 )
as FormattedCurPrice
FROM   product, cgprrel, prodprcs, prodsgp
WHERE  CGPRREL.cpcgnbr=$(cgrfnbr)
        and CGPRREL.cpmenbr=$(cgmenbr)
        and CGPRREL.cpprnbr=PRODUCT.prrfnbr
        and prpub=1
        and (($(NC_NOW) >= PRODPRCS.ppdeffs and $(NC_NOW) <= PRODPRCS.ppdefff) or
       (PRODPRCS.ppdeffs is null and PRODPRCS.ppdefff is null))
        and PRODSGP.psprnbr=PRODUCT.prrfnbr
        and PRODPRCS.pppmnbr=PRODUCT.prrfnbr
     and PRODPRCS.ppsgnbr=$(OHT_CODE)
     and PRODPRCS.ppmenbr=$(MerchantRefNum)
ORDER BY cpseqnbr ASC, pppre DESC
.
.
.
```

Figure 47. SQL statement of function DISPLAY_PRODUCT_SINGLEPRICE()

We added the table prodsgp, and some *where* conditions, such as and PRODPRCS.ppsgnbr=$(OHT_CODE).

This column, ppsgnbr, of the table, PRODPRCS, is the key to getting the correct price for a specific shopper group.

The $(OHT_CODE) variable is the result of two other functions that we included. Those functions are: GET_SHOPERGROUP and GET_CODEPRICE

The GET_SHOPERGROUP function gets the shopper group code from the table, SHADDR, using the variable, $(SESSION_ID), which has the Username of the shopper. If the shopper is not a registered shopper, the variable, $(SESSION_ID), has a code generated by the WCS. See Figure 48.

```
%function(dtw_odbc) GET_SHOPERGROUP() {
SELECT sanick, sashnbr
FROM   shaddr
WHERE  SHADDR.sanick='$(SESSION_ID)'
%REPORT{
    %ROW{
        @DTW_assign(OHT_USER, V_SASHNBR)
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_USER, NULL) %} :continue
        default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME)
        because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
 %}
%}
```

Figure 48. The GET_SHOPPERGROUP()function

You can see that, if a shopper group code exists the $(OHT_USER) variable gets this value; otherwise, if a shopper group does not exist (because it is not a registered user), the $(OHT_USER) variable gets a NULL value.

The GET_CODEPRICE function uses the variable, $(OHT_USER), to get the price code that belongs to the shopper group.

```
%function(dtw_odbc) GET_CODEPRICE() {
SELECT mcshnbr, mcsgnbr
FROM   mcustinfo
WHERE  MCUSTINFO.mcshnbr=$(OHT_USER)
%REPORT{
    %ROW{
        @DTW_assign(OHT_CODE, V_MCSGNBR)
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_CODE, NULL) %} :continue
        default:{<BR>Problem with GET_CODEPRICE() in file $(DTW_MACRO_FILENAME)
        because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
 %}
%}
```

Figure 49. The GET_CODEPRICE()function

Now, using the *MCUSTINFO* table, which has the relationship between the shoppers with shopper groups, the variable, $(OHT_CODE), gets the code of the shopper group, and, with this value, we can get to the correct product price for a specific shopper group.

Finally, we modify the HTML_REPORT section of the macro

```
%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{
.
.
.
<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment)
CELLPADDING=4 CELLSPACING=0>
<!--A HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/InterestItemAdd?product_rn=
$(prrfnbr)
&merchant_rn=$(MerchantRefNum)&WISHLISTSHOPPER_RN=$(SESSION_RN)
&url=http://$(HOST_NAME)/cgi-bin/ncommerce3/InterestItemDisplay">
<FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)">Add to my wishlist</FONT>
</A-->
@DISPLAY_PRODUCT_IMAGE()
@DISPLAY_PRODUCT_INFO()
%IF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor == NULL ||
ConvDivideFactor == NULL))
  @GET_SHOPERGROUP()
  %IF (OHT_USER == NULL)
       <TR><TD>@DISPLAY_PRODUCT_SINGLEPRICE_ISNULL()</TD></TR>
  %ELSE
       @GET_CODEPRICE()
       @GET_CODEPRICE()
        %IF (OHT_CODE == NULL)
           <TR><TD>@DISPLAY_PRODUCT_SINGLEPRICE_ISNULL()</TD></TR>
       %ELSE
           <TR><TD>@DISPLAY_PRODUCT_SINGLEPRICE()</TD></TR>
       %ENDIF
  %ENDIF
  %ELIF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor != NULL &&
   ConvDivideFactor != NULL))
       @DISPLAY_PRODUCT_DUALPRICE()
  %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor == NULL ||
   ConvDivideFactor == NULL))
       @DISPLAY_PRODUCT_SINGLEPRICE_RANGE()
       @DISPLAY_ITEMS_DROPDOWN()
  %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor != NULL &&
   ConvDivideFactor != NULL))
.
.
.
```

Figure 50. The new HTML_Report section

The next step is to make a new function to manage those cases when the shopper is not a registered user.

This new function is called in the last figure, and the name of the function is DISPLAY_PRODUCT_SINGLEPRICE_ISNULL.

This function is almost the same as the function, *DISPLAY_PRODUCT_SINGLEPRICE*, but, as we can see in Figure 51, we now compare the column, ppsgnbr, from the table, PRODPRCS, with the condition, is NULL, instead of the variable, $(OHT_CODE).

```
%function(dtw_odbc) DISPLAY_PRODUCT_SINGLEPRICE_ISNULL(){

    SELECT prnbr, ppprc, ppcur, pppre, ppdeffs, ppdefff,
           NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces),
'en_US', 1 ) as FormattedCurPrice
    FROM product, prodprcs, prodsgp
    WHERE prmenbr=$(MerchantRefNum)
                and prrfnbr=$(prrfnbr)
                and ppprnbr=$(prrfnbr)
          and PRODPRCS.ppsgnbr is NULL
                and PRODSGP.psprnbr=$(prrfnbr)
          and (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or
            (ppdeffs is null and ppdefff is null))
    ORDER BY pppre DESC
    %REPORT{
    @DTW_ASSIGN(ACTUALPRICE, "0")
    <TR BGCOLOR=$(HighLightColor2)>
        <TD ALIGN="center" COLSPAN=2>
    %ROW{
        %IF (ROW_NUM == "1")
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
            <B>$(LBL_PRODUCTPRICE) : </B>
            </FONT>
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
            $(V_FormattedCurPrice) $(CurDescription)</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
            @DTW_ASSIGN(ACTUALPRICE, V_FormattedCurPrice)
        %ENDIF
        %IF (V_pppre == "0" && ACTUALPRICE < V_FormattedCurPrice)
            <FONT FACE="$(BodyFontFace)" SIZE="-1">
            (<STRIKE>$(V_FormattedCurPrice) $(CurDescription)</STRIKE>)</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
        %ENDIF
%}
        </TD>
    </TR>
.
.
.
```

Figure 51. The DISPLAY_PRODUCT_SINGLEPRICE_ISNULL function

As in both cases, category and the product macros, we must proceed with the cat_catalog.d2w macro, which displays the initial page of the catalog with those featured products and the categories.

As in the cat_category.d2w and cat_product.d2w macros, when a store is created, the WCS gives a consecutive number for this store; so, in the hard

drive, we can see, on locations X:\IBM\NetCommerce3\macro\en_US (for Windows NT 4.0) and usr/lpp/NetCommerce3/macro/en_US (for AIX), a sub-directory named after the number of the store (for example, our store has the number 831).

On this subdirectory, we can find the macro, cat_product.d2w, which contains a few lines of code as shown in Figure 52.

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_catalog.d2w"
```

Figure 52. The macro cat_catalog.d2w

Here, the first two lines of the macro include another two environment macros: include.inc and theme.inc.

Both are located on the path, X:\IBM\NetCommerce3\macro\en_US\ (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/en_US/ (for AIX), on a subdirectory with the same number as the store (for example, our store has the number 831).

Those macros declare several environment variables, such as color of the background, text, buttons, and so on.

The last line is the most important because this line invokes the macro that really does the work.

This macro is located on the path X:\IBM\NetCommerce3\macro\common\ClassicStoreModel (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/common/ClassicStoreModel (for AIX).

This is a common macro. It means that if you make some changes on it, these changes can affect the behavior of the other stores.

To prevent this, we can make a copy of this macro with a name, such as cat_catalog<number of the store>.d2w (even in the same subdirectory) and make all the changes on it (for example, cat_catalog831.d2w).

The only change we must make on the macro, cat_catalog.d2w, is to change the name of the macro in the last line,
%include "ClassicStoreModel\cat_product.d2w", to
%include "ClassicStoreModel\cat_catalog831.d2w". See Figure 53.

```
%include "831\include.inc"
%include "831\theme.inc"
%include "ClassicStoreModel\cat_catalog831.d2w"
```

Figure 53. The macro cat_cataloc.d2w after the changes

Now, we can modify the macro, cat_catalog<number of the store>.d2w, located in X:\IBM\NetCommerce3\macro\common\ClassicStoreModel (for Windows NT 4.0) or usr/lpp/NetCommerce3/macro/common/ClassicStoreModel (for AIX).

If you see the report section of this macro, you can see that the macro has several functions to present the categories of information on the browser.

Those functions are:

- %function(dtw_odbc) DEBUG_FEATUREPRODUCTS()
- %function(dtw_odbc) COUNT_FEATUREPRODUCTS()
- %function(dtw_odbc) CALC_RANDOMFEATUREDPRODUCTROW
- %function(dtw_odbc) GET_FEATUREPRODUCT()
- %function(dtw_odbc) GET_LATESTPRODUCT()
- %function(dtw_odbc) GET_SINGLEPRICE_FEATUREPRODUCT()
- %function(dtw_odbc) GET_SINGLEPRICE_FEATUREPRODUCT_ISNULL()
- %function(dtw_odbc) GET_DUALPRICE_FEATUREPRODUCT()
- %function(dtw_odbc) DISPLAY_CATEGORIES()
- %function(dtw_odbc) DISPLAY_PRODUCT_LIST_DUALPRICE()
- %function(dtw_odbc) DISPLAY_PRODUCT_LIST_SINGLEPRICE()

In Figure 54 on page 91, we can see how some of the most important functions in the Report Section are called.

```
%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{
<HTML>
@DTW_TIME("S", SEED)
@DTW_SUBSTR(SEED, @DTW_rLENGTH(SEED), "1", SEED)
@COUNT_FEATUREPRODUCTS()
%IF(RANDOM_ENDBOUND > "0")
    @CALC_RANDOMFEATUREDPRODUCTROW()
    @GET_FEATUREPRODUCT()
%ELIF (RANDOM_ENDBOUND == "0")
    @DTW_ASSIGN(RANDOM_ROWNUM, "1")
    @GET_FEATUREPRODUCT()
%ELSE
    @GET_LATESTPRODUCT()
%ENDIF
%IF (ConvMultiplyFactor == NULL || ConvDivideFactor == NULL)
    @GET_SINGLEPRICE_FEATUREPRODUCT()
%ELSE
    @GET_DUALPRICE_FEATUREPRODUCT()
%ENDIF
<HEAD>
    <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1999 01:01:01 GMT">
</HEAD>
<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)" TEXT="$(TextCol)"
LINK="$(LinkCol)" VLINK="$(VLinkCol)" ALINK="$(ALinkCol)">
@DTW_ADD(MerchantRefNum, "0", "2" , SecurityCheck)
@DTW_ASSIGN(NOCATEGORIESINCATEGORY, "NO")
@DTW_ASSIGN(NOPRODUCTSINCATEGORY, "NO")
<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
    <TR>
        <TD ALIGN=$(TitleAlignment) VALIGN="center">
        %IF (PAGE == "LOGON")
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
            SIZE=$(TitleFontSize)><B>Welcome back $(SESSION_ID)!</B></FONT>
        %ELSE
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(TXT_TITLE_CATALOG)</B></FONT>
        %ENDIF
```

Figure 54. Important section of the cat_catalog< store number>.d2w file

The function we will modify is GET_SINGLEPRICE_FEATUREPRODUCT() because this is the function that displays the price of the product chosen by the customer from the product list.

In Figure 55, we show the modified GET_SINGLEPRICE_FEATUREPRODUCT() function.

```
%FUNCTION(dtw_odbc) GET_SINGLEPRICE_FEATUREPRODUCT(){
    SELECT pravdate, prfull, prsdesc, prldesc1, ppprc, prrfnbr,
       NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), '', 1 )
as FormattedFeatureCurPrice
    FROM product, prodprcs
    where   prmenbr=$(MerchantRefNum)
        and prrfnbr=$(FEATURE_PRRFNBR)
        and ppmenbr=prmenbr
        and pppmnbr=prrfnbr
        and (($(NC_NOW)>= ppdeffs and $(NC_NOW)<= ppdefff) or (ppdeffs is null and
       ppdefff is null))
     and PRODPRCS.ppsgnbr=$(OHT_CODE)

    ORDER BY pppre ASC

    %REPORT{

       @DTW_ASSIGN(FEATURE_PRICE, V_FormattedFeatureCurPrice)
    %}
    %MESSAGE{
       100:{%} :continue
       default:{<BR>Problem with GET_SINGLEPRICE_FEATUREPRODUCT()
       in file $(DTW_MACRO_FILENAME) because of error $(RETURN_CODE).
       <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}
```

Figure 55. The modified GET_SINGLEPRICE_FEATUREPRODUCT() function

We added the table, prodsgp, and some where conditions, such as *and PRODPRCS.ppsgnbr=$(OHT_CODE)*.

The ppsgnbr column of the PRODPRCS table is the key to getting the correct price for a specific shopper group.

The $(OHT_CODE) variable is the result from the other two functions that we included. Those functions are GET_SHOPERGROUP and GET_CODEPRICE.

The GET_SHOPERGROUP function, shown in Figure 56 on page 93, gets the shopper group code from the SHADDR table using the $(SESSION_ID) variable, which has the Username of the shopper. If the shopper is not a registered shopper, the $(SESSION_ID) variable has a code generated by the WCS.

```
%function(dtw_odbc) GET_SHOPERGROUP() {
SELECT sanick, sashnbr
FROM   shaddr
WHERE  SHADDR.sanick='$(SESSION_ID)'
%REPORT{
    %ROW{
        @DTW_assign(OHT_USER, V_SASHNBR)
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_USER, NULL) %} :continue
        default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME)
        because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}
```

Figure 56. The GET_SHOPPERGROUP()function

You can see that, if a shopper group code exists the $(OHT_USER) variable gets this value; otherwise, if a shopper group does not exist (because it is not a registered user), the $(OHT_USER) variable gets a NULL value.

The GET_CODEPRICE function uses the $(OHT_USER) variable to get the code of the price that belongs to the shopper group.

```
%function(dtw_odbc) GET_CODEPRICE() {
SELECT mcshnbr, mcsgnbr
FROM   mcustinfo
WHERE  MCUSTINFO.mcshnbr=$(OHT_USER)
%REPORT{
    %ROW{
        @DTW_assign(OHT_CODE, V_MCSGNBR)
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_CODE, NULL) %} :continue
        default:{<BR>Problem with GET_CODEPRICE() in file $(DTW_MACRO_FILENAME)
        because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}
```

Figure 57. The GET_CODEPRICE()function

Now, using the *MCUSTINFO* table, which has the relationship between the shoppers and shopper groups, the $(OHT_CODE) variable gets the code of the shopper group, and, with this value, we can to get the correct product price for a specific shopper group.

Finally, we modify the HTML_REPORT section of the macro in Figure 58.

```
%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{

<HTML>

@DTW_TIME("S", SEED)
@DTW_SUBSTR(SEED, @DTW_rLENGTH(SEED), "1", SEED)

@COUNT_FEATUREPRODUCTS()

%IF(RANDOM_ENDBOUND > "0")
    @CALC_RANDOMFEATUREDPRODUCTROW()
    @GET_FEATUREPRODUCT()
%ELIF (RANDOM_ENDBOUND == "0")
    @DTW_ASSIGN(RANDOM_ROWNUM, "1")
    @GET_FEATUREPRODUCT()
%ELSE
    @GET_LATESTPRODUCT()
%ENDIF

%IF (ConvMultiplyFactor == NULL || ConvDivideFactor == NULL)
     @GET_SHOPERGROUP()
     %IF (OHT_USER == NULL)
        @GET_SINGLEPRICE_FEATUREPRODUCT_ISNULL()
    %ELSE
        @GET_CODEPRICE()
         %IF (OHT_CODE == NULL)
            @GET_SINGLEPRICE_FEATUREPRODUCT_ISNULL()
        %ELSE
            @GET_SINGLEPRICE_FEATUREPRODUCT()
        %ENDIF
    %ENDIF
%ELSE
    @GET_DUALPRICE_FEATUREPRODUCT()
%ENDIF

<HEAD>
    <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1999 01:01:01 GMT">
</HEAD>

<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)" TEXT="$(TextCol)"
LINK="$(LinkCol)" VLINK="$(VLinkCol)" ALINK="$(ALinkCol)">
.
.
.
```

Figure 58. Part of the new HTML_Report section

The next step is to make a new function that manages those cases when the shopper is not a registered user.

This new function is called in the last figure, and the name of the function is GET_SINGLEPRICE_FEATUREPRODUCT_ISNULL().

This function is almost the same as the GET_SINGLEPRICE_FEATUREPRODUCT() function, but, as we can see in Figure 59, we now compare the ppsgnbr column from the PRODPRCS table with the *is NULL* condition instead of the $(OHT_CODE) variable.

```
%FUNCTION(dtw_odbc) GET_SINGLEPRICE_FEATUREPRODUCT_ISNULL(){
    SELECT pravdate, prfull, prsdesc, prldesc1, ppprc, prrfnbr,
       NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), '', 1 )
as FormattedFeatureCurPrice
    FROM product, prodprcs
    where   prmenbr=$(MerchantRefNum)
        and prrfnbr=$(FEATURE_PRRFNBR)
        and ppmenbr=prmenbr
        and pppmnbr=prrfnbr
        and (($(NC_NOW)>= ppdeffs and $(NC_NOW)<= ppdefff) or
       (ppdeffs is nulland (ppdefff is null))and PRODPRCS.ppsgnbr is NULL

    ORDER BY pppre ASC

    %REPORT{

       @DTW_ASSIGN(FEATURE_PRICE, V_FormattedFeatureCurPrice)
    %}
    %MESSAGE{
       100:{%} :continue
       default:{<BR>Problem with GET_SINGLEPRICE_FEATUREPRODUCT()
in file $(DTW_MACRO_FILENAME) because of error $(RETURN_CODE).
<BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}
```

Figure 59. The GET_SINGLEPRICE_FEATUREPRODUCT_ISNULL function

5.1.2.2 Customization of the merchant tool

Now, we must work on the customization of the merchant tool because, as we can see, this tool does not offer any feature to create, manage, or add registered shoppers to a shopper group.

We split this customization into two parts:

1. Customization of the navigation menu
2. Creating the macros that manage the databases

The first part consists of several steps to show the merchant this new feature in the merchant tool, and the second part consists of creating those macros

that manage the database to create, update, and manage the shopper groups.

Note

Remember that those changes must be only for the Classic Store Model, and not for the Express Store model

Customization of the navigation menu

This menu consist of two main parts: An *.xml file named navigation.xml and located in the path, /usr/lpp/NetCommerce3/Tools/xml/nchs/mtool/advanced/ (for AIX) or X:\Ibm\NetCommerce3\Tools\xml\nchs\mtool\advanced (for Windows NT 4.0), and the file navigationNLS.properties located in the compiled *.jar file named nchs.jar. This *.jar file is located in the path /usr/lpp/NetCommerce3/Tools/lib (for AIX) or X:\Ibm\NetCommerce3\Tools\lib (for Windows NT 4.0).

The first one is an XML file that contains all the instructions for showing the menu bar on the left side frame of the merchant tool.

We must add some lines to create a new category (node), named Shopper Groups, and two options (sub-nodes), named create shopper groups and manage shopper groups.

In Figure 60 on page 97, we can see part of the navigation.xml file.

```
<?xml version="1.0"?>
<tree resourceBundle="nchs.navigationNLS">
  <node name="root"
        image="/NCTools/images/clear.gif"
        openImage="/NCTools/images/clear.gif"
         users="siteAdmin storeAdmin catalogAdmin orderAdmin" >
    <node name="viewStore"
          url="/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.StoreLink"
          image="/NCTools/images/bullet_blank.gif"
           users="siteAdmin storeAdmin catalogAdmin orderAdmin" />
    <node name="gettingStarted"
          image="/NCTools/images/clear.gif"
          openImage="/NCTools/images/clear.gif"
          users="siteAdmin storeAdmin catalogAdmin orderAdmin" >
      <node name="quickStart"
            url="/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.GetStarted"
            image="/NCTools/images/clear.gif"
            users="siteAdmin storeAdmin catalogAdmin orderAdmin" />
      <node name="getMerchantGuide"
            url="/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.MerGuide"
            image="/NCTools/images/clear.gif"
              users="siteAdmin storeAdmin catalogAdmin orderAdmin" />
.
     </node>
    <node name="shopperGroups"
          image="/NCTools/images/clear.gif"
          openImage="/NCTools/images/clear.gif"
          users="siteAdmin storeAdmin orderAdmin" >
    <node name="createShopperGroups"
      url="http://www.themall.com/cgi-bin/ncommerce3/ExecMacro/creagrou.d2w/report"
      image="/NCTools/images/clear.gif"
      users="siteAdmin storeAdmin" />
    <node name="addShopperstoGroups"
      url="http://www.themall.com/cgi-bin/ncommerce3/ExecMacro/addshgru.d2w/report""
      image="/NCTools/images/clear.gif"
      users="siteAdmin storeAdmin" />
    <node name="GroupstoPrices"
      url="http://www.themall.com/cgi-bin/ncommerce3/ExecMacro/addprgru.d2w/report"
      image="/NCTools/images/clear.gif"
      users="siteAdmin storeAdmin" />
     </node>
</node>
</tree>
```

Figure 60. The navigation.xml file

We can see the statement, <tree resourceBundle="nchs.navigationNLS">, on the second line; this indicates which compiled file (*nchs*.jar) and which properties file (*navigationNLS*.properties) we must customize to complete this part. We recommend creating a new sub-directory, called WORK, and making a copy of the nchs.jar file on it. We must extract all the contents of the nchs.jar (with sub-directories) on the same directory, and then to open the file, navigationNLS.properties, located in the new path

/usr/work/com/ibm/commerce/tools/nchs/properties (on AIX) or X:\work\com\ibm\commerce\tools\nchs\properties (on Windows NT 4.0)

With a text editor, we must add some lines as shown in Figure 61.

```
shopperGroups=Shopper Groups
createShopperGroups=create shopper groups
addShopperstoGroups=Add shoppers to a group
GroupstoPrices=Assign prices to a group
```

Figure 61. The new added lines on navigationNLS.properties file

Now, save the file and recompile all the *.jar files.

To do this, type `jar cvf new.jar com/*` in the path X:\work (for Windows NT 4.0) or /usr/work (for AIX).

We must copy this new.jar file into the path /usr/lpp/NetCommerce3/Tools/lib (for AIX) or X:\Ibm\NetCommerce3\Tools\lib (for Windows NT 4.0).

Then, we must stop the WCS (WCS Servlet Service for Windows NT 4.0), rename the old nchs.jar file, rename the new.jar file as nchs.jar, and, finally, restart the WCS.

Create the macros that manage the databases
First of all, we must create those shopper groups that will handle the registered shoppers in the merchant's store.

As we saw in the customization of the navigation menu part, in the second sub-node (<node name="createShopperGroups"), we can see the URL `http://www.themall.com/cgi-bin/ncommerce3/ExecMacro/creagrou.d2w/report` located in the path, /usr/lpp/NetCommerce3/Tools/xml/nchs/mtool/advanced/ (for AIX) or X:\Ibm\NetCommerce3\Tools\xml\nchs\mtool\advanced (for Windows NT 4.0).

This is called to a macro, named creagrou.d2w, which asks for the new shopper group data (the name of the shopper group and a short description) and then proceeds to create the new shopper group. The macro, creagrou.d2w, is listed in Appendix B.4, "Macro file creagru.d2w" on page 272.

The main purpose of this macro is to manage the SHOPGRP table, which describes the shopper groups that are defined for each store. Each row of this table contains information on one shopper group.

Another table to manage is the KEYS table, which contains the current maximum values of the primary keys of the tables:

- ORDERS
- SHIPTO
- SHOPPER
- SHADDR
- TASKS
- TAXCGRY
- SHIPMODE, STRCGRY
- MERCHANT
- MSHIPMODE
- ACC_GROUP
- PRSPCODE
- SHIPPING
- SHOPGRP
- PRODUCT
- PRODPRCS
- CATEGORY
- SCALE
- DISCCODE
- STAGLOG
- ACC_MODE

When a merchant executes the macro using the merchant tool, the result is something like that shown in Figure 62 on page 100.

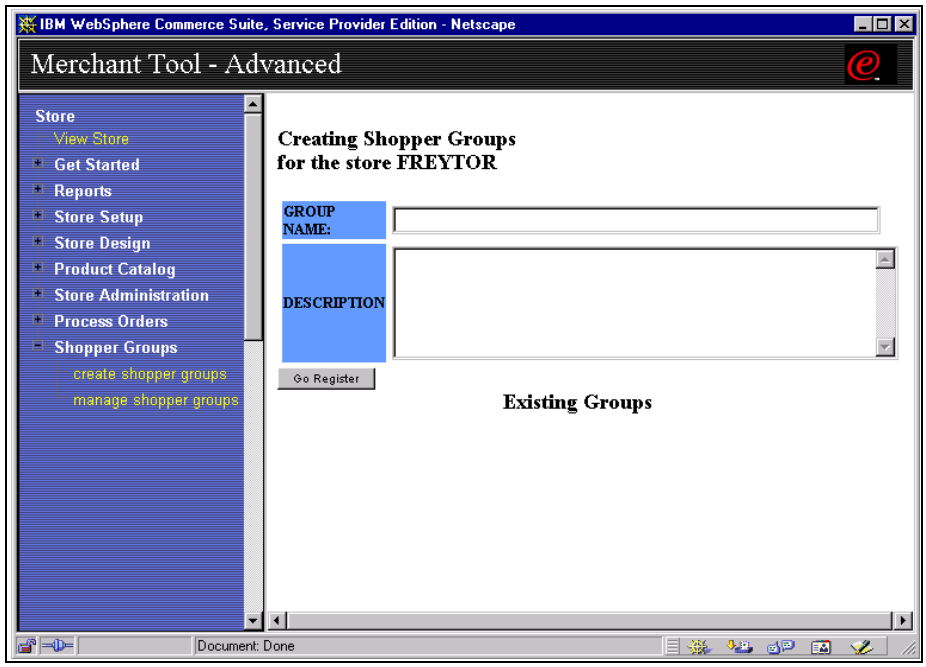


Figure 62. The screen of the creagrou.d2w macro

The merchant simply needs to type the name of the new shopper group in the field called Group Name:, a short description of this shopper group in the field called Description:, and, finally, select **Go Register**.

Once the shopper groups have been created, the next step is to assign those registered shoppers that do not yet belong to a shopper group.

To solve this, we create a macro called addshgru.d2w. The macro is listed in Appendix B.5, “Macro file addshgru.d2w” on page 276.

With this new feature, the merchant can choose between those registered shoppers with no shopper group, choose a group previously created, and, finally, see the shoppers contained in the chosen group.

In Figure 63 on page 101, we show the graphic interface that the merchant can see.

Figure 63. The screen of the addshgru.d2w macro

Finally, the macro that assigns prices to products for each shopper group, addprgru.d2w, is created. The macro, addprgru.d2w, is listed in Appendix B.6, “Macro addprgru.d2w” on page 281.

In Figure 64 on page 102, we can see the first page that the addprgru.d2w macro shows.

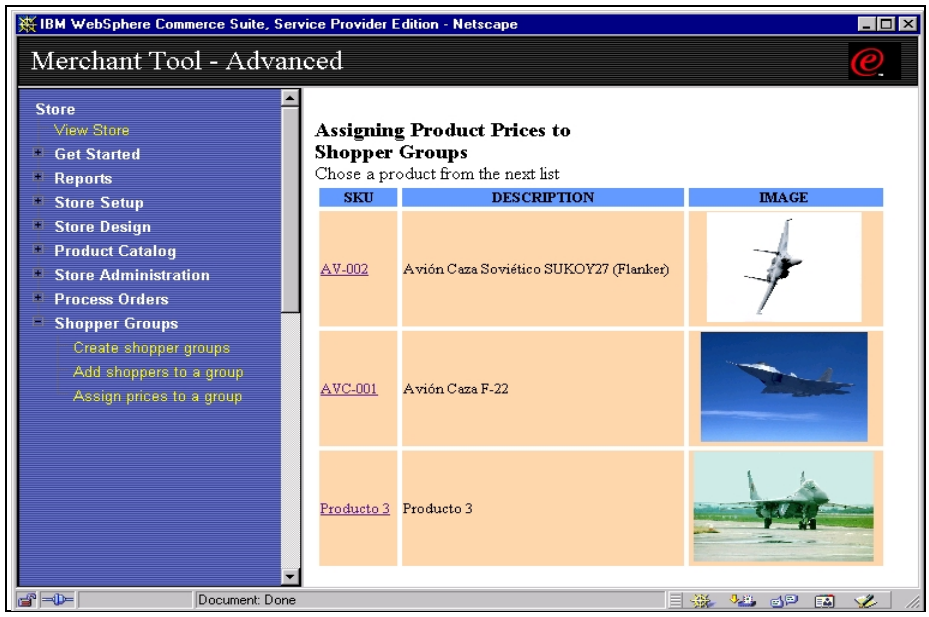


Figure 64. The screen of the addprgru.d2w macro

Chapter 6. Cross-sell and up-sell

WebSphere Commerce Suite Service Provider Edition (WCS-SPE) provides a feature to customize products displayed in an online store to follow the two useful business strategies of selling, namely, cross-selling and up-selling of products. The objective of this section is to create a store that has cross-sell and up-sell features. The intended audience for this chapter is e-business Service Providers and store managers of merchants who own stores and need to customize them for cross-selling and up-selling.

6.1 What is cross-selling and up-selling

Cross-selling refers to suggesting a product that compliments what the shopper has already chosen. For example, if the shopper has selected corn flakes, you can suggest a carton of milk to go with the cereal.

In a real store scenario, merchants or retailers often place related merchandise in one area. Sales clerks are always on the lookout for an opportunity to sell a more expensive product or an accessory product if it has a higher profit margin. For example, if a shopper is looking at gloves, he or she may also be willing to purchase a scarf that goes with it, or, after a shopper buys a pair of shoes, he or she may consider purchasing a shoe-horn or shoeshine kit. Several relationship types between the products are possible including compatible accessory, cross-sell, or up-sell items. Let us look at a definition for each of these relationships as shown in Table 2.

Table 2. Product relationship types

Product relationship type	Description
Up-sell	Related items to be promoted to entice shopper up the price scale
Cross-sell	Related items to be promoted on the same price scale
Compatible	Compatible items to be promoted as accessories

In an online store, when a product page is displayed, there can be a hyperlink to the related product. Once the shopper is interested in the related product, he or she clicks on the link that takes him to another window where the text, *May we suggest*, appears at the top of the page for cross-selling, or *This is better* appears for up-selling, or *You might also be interested in* appears for compatible accessories. A thumbnail image of the suggested product, a description, a price, and a button to add the product to the shopping cart can

also be displayed in the window. The shopper can add the product to the shopping cart if they wish to buy the product, or they may close the window to continue shopping.

6.2 Customization techniques

Some customization technique that can be followed to get the features of cross-sell and up-sell of products are:

1. Create tables for product relationships and import data through a mass import utility.
2. Import data through a browser-based mass import utility.
3. Customize the Merchant tool catalog editor to create the relationship between the products.

6.2.1 Creating product relationship by mass import utility

Creating a product relationship between the products with the mass import utility works fine for cross-selling or up-selling multiple products. This is done in the following way:

1. Create two tables: One for defining the product relationship types and another for creating the exact relationships between the products.
2. Customize the Net.Data macro that displays the product page.

To achieve this functionality, a reference application, related.exe (54 KB), can be downloaded. The application is available on the Websphere Commerce Server reference site, which, at the time of this writing, was found in the following URL:

`www-4.ibm.com/software/webservers/commerce/community/process/refapps.html`

The application identifies related or compatible products with IBM WebSphere Commerce Suite and also introduces two new tables to represent product relationships, such as compatible accessory, cross-sell, or up-sell items. You can skip this section by downloading the application and following the instructions as given in the accessorizer.pdf document, or you can perform the steps provided in the next section to create tables and customize Net.Data macros. The user interface in this application package is based on the Net.Commerce sample Metropolitan Mall (demomall). Changing this application package to work with your own interface should be relatively straightforward.

6.2.1.1 Creating product relationship tables in demomall database

This section describes the definition of two new tables in the demomall database and registering them with the Net.Commerce *Keys* manager. A database connection to demomall is needed to update the current database. Perform the following steps:

1. Log on as a DB2 administrator and open a DB2 command prompt.
2. Connect to the database by issuing the following query:

   ```
   connect to demomall
   ```

3. Create the product relationship type table, PRRELTYPE. The technical details are listed in Table 3.

Table 3. PRRELTYPE: Product relationship type table details

Column name	Column type	Description
prtrfnbr	INTEGER NOT NULL	The reference number for the Product Relationship Type. This is the primary key.
prtname	CHAR(30) NOT NULL	The name of the relationship type.
prtdesc	CHAR(254)	The description of the relationship type.
prtfield1	INTEGER	Reserved for merchant customization.
prtfield2	INTEGER	Reserved for merchant customization.
prtfield3	CHAR(254)	Reserved for merchant customization.

The following set of queries creates the table:

```
drop index ui_prt1;
drop table PRRELTYPE;
create table PRRELTYPE(prtrfnbr integer not null, prtname char(30) not
null, prtdesc char(254), prtfield1 integer, prtfield2 integer, prtfield3
char(254), constraint p_prreltype primary key (prtrfnbr));
create unique index ui_prt1 on PRRELTYPE (prtname);
```

4. Create a table, PRPRREL, that creates the exact relationship between the product. The technical details of the table are as follows:

Table 4. PRPRREL: Product-to-product relationship table details

Column name	Column type	Description
plrfnbr	INTEGER NOT NULL	The reference number for the Product Relationship. This is the primary key.
plmenbr	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references the MERFNBR column in the MERCHANT table.
plprtnbr	INTEGER NOT NULL	The reference number of the Product Relationship Type. This is a foreign key that references the PRTRFNBR column in the PRRELTYPE table.
plprnbr	INTEGER NOT NULL	The Product Reference Number of the base product. This is a foreign key that references the PRRFNBR column in the PRODUCT table.
plrelprnbr	INTEGER NOT NULL	The Product Reference Number of the related product. This is a foreign key that references the PRRFNBR column in the PRODUCT table.
plqty	INTEGER	Recommended quantity of the related product, for example, two batteries per flashlight.
plseqnbr	INTEGER	A sequence number for the display.
plsgnbr	INTEGER	The Shopper Group Reference Number. This is a foreign key that references the SGRFNBR column in the SHOPGRP table.
plpub	INTEGER	Publish flag. Use 1 to indicate that the relationship should be published.
plfield1	INTEGER	Reserved for merchant customization.
plfield2	CHAR(254)	Reserved for merchant customization.

The following set of queries creates the table with the required index:

```
drop index ui_prprrel;
drop index i_prprrel1;
drop index i_prprrel2;
drop index i_prprrel3;
drop table prprrel;
create table prprrel (plrfnbr integer not null, plmenbr integer not
null, plprtnbr integer not null, plprnbr integer not null, plrelprnbr
```

```
integer not null, plqty integer, plseqnbr float, plsgnbr integer, plpub
smallint, plfield1 integer, plfield2 char(254), constraint   p_prprrel
primary key (plrfnbr), constraint fme_prprrel foreign key (plmenbr)
references merchant (merfnbr) on delete cascade, constraint
fpr_prprrel1 foreign key (plprnbr) references product (prrfnbr) on
delete cascade, constraint   fpr_prprrel2 foreign key (plrelprnbr)
references product (prrfnbr) on delete cascade, constraint   fsg_prprrel
foreign key (plsgnbr) references shopgrp (sgrfnbr) on delete cascade,
constraint   fprl_pl foreign key (plprtnbr) references PRRELTYPE
(prtrfnbr) on delete cascade);
create unique index ui_prprrel on prprrel (PLMENBR, PLPRNBR, PLRELPRNBR,
PLSGNBR, PLPRTNBR);
create index i_prprrel1 on prprrel (plmenbr);
create index i_prprrel2 on prprrel (plprnbr);
create index i_prprrel3 on prprrel (plrelprnbr);
```

5. Register with the key manager of the Net.Commerce database. The query to do that is as follows:

```
delete from KEYS where keytable = 'prprrel';
delete from KEYS where keytable = 'prreltype';
insert into KEYS (keyrfnbr, keytable, keycolumn, keymaxid) values
((select max(keyrfnbr) + 1 from KEYS), 'prprrel', 'plrfnbr', 0);
insert into KEYS (keyrfnbr, keytable, keycolumn, keymaxid) values
((select max(keyrfnbr) + 1 from KEYS), 'prreltype', 'prtrfnbr', 0);
```

6. The next step is to define the relationship types in the PRRELTYPE table. There are three predefined relationship types in this solution as shown in Table 1 on page 60. To populate the prreltype table with the predefined relationship, the following set of queries is used:

```
delete from PRRELTYPE;
insert into PRRELTYPE (prtrfnbr, prtname, prtdesc) values
(1,'Compatible', 'Compatible items to be promoted as accessories');
insert into PRRELTYPE (prtrfnbr, prtname, prtdesc) values
(2,'Up-sell','Related items to be promoted to entice shopper up the
price scale');
insert into PRRELTYPE (prtrfnbr, prtname, prtdesc) values (3,
'Cross-sell', 'Related items to be promoted on the same price scale');
update keys set (keymaxid) = (select max(prtrfnbr) from prreltype) where
keytable = 'prreltype';
```

6.2.1.2 Populating the product-to-product relationship table

It is required to populate the product-to-product relationship table (PRPRREL) with the data. To do this, let us consider one example and proceed. The Metropolitan mall example shipped with the Net.Commerce product can be used. It contains several store examples. One is called 6th Avenue Department Store. This store is programmed to sell hardware, such

as hand tools and gardening tools, computer related products, such as computer hardware and software, and fashion-related items. We will consider one example, say, Computer Hardware, which is used to sell personal computers and printers.

Let us define the scope of the product to be sold and the related products to be shown as cross-sell product or up-sell product. Suppose a shopper wishes to buy an IBM Aptiva M53 type of PC. When the product page for IBM Aptiva M53 is loaded, there might be some hyperlinks to related products also displayed on the Web page. The IBM Aptiva M40 type of PC is better than M53; so, it can be up-sold, and software products, such as PC DOS 7, OS/2 Warp CD Pak, and Kid Riffs can be cross-sold. A display about compatible accessories for the PC, such as Winwriter 150 and Lexmark Optra E, can also be sold. This is shown in Figure 65 on page 109.

Figure 65. Product page with cross-sell and up-sell features

To achieve this feature, the data has to be uploaded in the prprrel table in the database. That is, for M53 types of PCs, the up-sold product is M40 and the cross-sold products are PC DOS 7, OS/2 Warp CD, and Kid Riffs, and the compatible accessories products are Winwriter 150 and lexmark Optra E. The easiest way to define this relationship between products in the database would be (if the number of products is very high) to use a mass import utility method. Perform the following steps:

1. Create a file, called sample_prprrel.mpt, in the text editor, and copy the following code into the file. This file has the actual data that relates the products:

```
#VERSION;32
#STORE;6ixth Avenue
#COLUMNDELIMITER;|
#PRPRREL|1|1|Compatible||S2168M53E14||P90Pt900E
#PRPRREL|1|1|Compatible||S2168M53E14||P182W150E
#PRPRREL|1|1|Cross-sell||S2168M53E14||083G9302
#PRPRREL|1|1|Cross-sell||S2168M53E14||O83G9303
#PRPRREL|1|1|Cross-sell||S2168M53E14||E2456841
#PRPRREL|1|1|Cross-sell||S2168M53E14||E2456842
#PRPRREL|1|1|Up-sell||S2168M53E14||03245341
```

   The creation of the mass import file is an important step for uploading the actual product data. The details of the PRPRREL table in Table 4 on page 106 relate to the above mass import file using “|” as the delimiter, and the file has the syntax described in Table 5.

Table 5. Details of mass import file

Value in the Mass import file	Corresponding element in the prprrel table	Description
#PRPRREL	name of the table	Name of the table
1	plpub	whether to publish the product or not
1	plqty	No. of quantity related with the product
Compatible	prtname	product relationship type name in prreltype table
Blank	plsgnbr	Shopper group reference number
S2168M53E14	plprnbr	1st product’s SKU number
Blank	plseq	Sequence number
P90Pt900E	plrelprnbr	2nd product’s SKU number

   Create a file called prprrel.ini in the text editor and copy the code given in the Appendix C.3, “Definition file of mass import utility (prprrel.ini)” on page 297.This file contains the definition of mass import file and the tables getting updated. You can also refer to the default definition file, massimpt.db2.ini, of the mass import utility in the /usr/lpp/NetCommerce3/bin directory. Since we have two new tables to be updated, the default file will differ from the prprrel.ini file with the following code that defines the syntax for two new tables:

```
#VERSION;32
#TABLE;prreltype
#COLUMNS;prtrfnbr;prtname;prtdesc;prtfield1;prtfield2;prtfield3
#UI;prtname
#REFNUM;prtrfnbr

#TABLE;prprrel
#COLUMNS;plrfnbr;plmenbr;plpub;plqty;plprtnbr;plsgnbr;plprnbr;plseqnbr;
plrelprnbr
#UI;plmenbr;plprnbr;plrelprnbr;plsgnbr;plprtnbr
#CONSTTOKEN;plmenbr;merchant
#REFNUM;plrfnbr
#FORREF;prprrel;plprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prprrel;plrelprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prprrel;plprtnbr;prreltype;prtrfnbr;prtname
```

2. connect to the demomall database by issuing the following query from the DB2 Command prompt:
   ```
   connect to demomall
   ```
3. Mass import the file issuing the following command from the DB2 command prompt. Ensure that you are in the same directory where the mass import file, sample_prprrel.mpt, and the definition file, prprrel.ini, exist.
   ```
   massimpt -infile sample_prprrel.mpt -db demomall -log prprrel.log
   -inputformat v32 -deffile prprrel.ini
   ```
4. Check the prprel.log.log file for any errors and the result of the database update.

In this way, the product-to-product relationship table (prprrel) has been populated with the required data using the mass import utility.

6.2.1.3 Customizing product display Net.Data macro

The next step is to customize the product display page that displays the *IBM Aptiva M53* product page. First, you need to know which macro is responsible for displaying the product page. This is done with the following method.

1. Right-click on the product page, and view the Frame info. If the page uses an `ExecMacro` command, the name of the macro is displayed in the URL. Find the exact macro that displays the page in your system. It is usually identified by the directory in which it resides. The directory name will be the merchant reference number. There may be many macros with the same name for various stores created in the system.

2. If the `ProductDisplay` command is used instead of ExecMacro, this command sets the PROD_DSP view task to display the product based on the merchant number and reference number. Use the merchant number field to filter the results from the macros or prodsgp tables to find out the exact macro and the path to identify where it is found in the file system.

In this case, tempcomp.d2w is the macro for the merchant reference number, 2066, and product number, 13, and it is available in the following directory in the AIX system:
/usr/lpp/NetCommerce3/macro/en_US/product
Since this macro is used by many stores to display the product pages containing computer products, you must make a copy of this file and call it tempcomp_crup.d2w to represent cross-selling and up-selling. Also, make one more copy of the same file and call it tempcomp_crup_win.d2w. This file is used to display the related product page in the new window. Perform the following steps to customize the product page to list cross-sell and up-sell products:

1. Open the tempcomp_crup.d2w file in the text editor.
2. Copy the following function as shown in Figure 66 on page 113 in the function definition block along with other database functions.

```
%{Cross sell, Up sell function starts here%}

%function(dtw_odbc) Sell(IN l_prtname){
 select plrelprnbr, prrfnbr, prnbr, prsdesc
 from prprrel, prreltype, product
 where
 plpub = 1 and
 prpub = 1 and
 prrfnbr = plrelprnbr and
 plprnbr = $(prrfnbr) and
 (plsgnbr is NULL OR plsgnbr=$(GroupNumber)) and
 plprtnbr = prtrfnbr and
 prtname = '$(l_prtname)'


%REPORT{
%ROW{
       @DTW_ASSIGN(prrelprnbr,$(V_prrfnbr))
     <tr>
 <td><a href=#
onClick=ShowSell($(prrelprnbr))>$(V_prnbr)</a> </td><td>$(V_prsdesc)</td>
<br>
     </tr>
%}
%}
%MESSAGE{100:{%}:continue%}
%}
%{Cross sell, Up sell function ends here%}
```

Figure 66. Cross-sell and up-sell functions

3. After calling the SQL18() function and before displaying the shopping cart button, copy the following section, as shown in Figure 67 on page 114, into

the HTML_REPORT section of the macro.

```
<!-- Cross Selling, Up selling code starts here -->

<table CELLSPACING=0 CELLPADDING=0 BORDER=0>

    <tr>
        <th colspan="2" align="left">
        This is better (Upsell)...:
        </th>
    </tr>

    @Sell("Up-sell")

    <tr>
        <th colspan="2" align="left">
        May we suggest (Cross-sell)...:
        </th>
    </tr>

    @Sell("Cross-sell")

    <tr>
        <th colspan="2" align="left">
        Compatible accessories:
        </th>
    </tr>

         @Sell("Compatible")

<SCRIPT LANGUAGE="javascript">

   function ShowSell(prrelprnbr)
   {
window.open("http://$(HOST_NAME)/cgi-bin/ncommerce3/ExecMacro/product/tempco
mp_crup_win.d2w/report?prrfnbr="+prrelprnbr+"&prmenbr=$(prmenbr)","sellwindo
w","toolbar=no,location=no,directories=no,status=no,menubar=no,resizable=yes
,width300,height=500,prompt=no,scrollbars=1,setTimeOut=no");
}
   </SCRIPT>

</table>

<!-- Cross Selling, Up selling code ends here -->
```

Figure 67. Code displaying cross or up sold products

4. Open the second macro, tempcomp_crup_win.d2w, in a text editor. This macro is used to display the related product page. Refer to Figure 67, the Java script function opens up a new window with the content as available in the tempcomp_crup_win.d2w macro. Now, it is necessary to add the functionality of closing the window and returning to the main window once the product is added to the shopping cart, or close the window before

proceeding. This message is added in the HTML_REPORT section of the macro.

To close the window, press the **Add to Shopping Cart** button; this will call the `InterestItemAdd` command and display the shopping cart page using the `InterestItemDisplay` command passed as a hidden variable in the form element. Now, our requirement is to close the window instead of displaying the shopping cart in the new page, but the item added to the shopping cart must be added before closing the window. The code, InterestItemDisplay, is passed in a hidden variable. An `ExecMacro` command is called to execute a macro called `closewin.d2w`, which does nothing but close the window. The code is shown in Figure 68.

```
<!-- Content of tempcomp_crup_win.d2w file which calls the closewin.d2w
macro to close the window -->
<INPUT TYPE=HIDDEN NAME="url"
VALUE="/cgi-bin/ncommerce3/ExecMacro/product/close win.d2w/report">
<!-- INPUT TYPE=HIDDEN NAME="url"
VALUE="/cgi-bin/ncommerce3/InterestItemDisplay" -->
<!-- Content of closewin.d2w - to be copied in a new file called
closewin.d2w-->

%HTML_REPORT {
<HTML>
 <HEAD>
 <TITLE>
   Close Window
 </TITLE>
 </HEAD>
 <BODY onLoad="window.close()">
 </BODY>
</HTML>
%}
```

Figure 68. Code for closing the new window

5. These steps will display a page similar to the one shown in Figure 65 on page 109. To test the macro, enter the following URL in your browser:
   `http://<host>/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=13&prmenbr=2066`
6. Click on the product that is displayed for cross-selling. For example, click on **0835903 PC DOS 7**. This will pop up a new window with the details of the product. This window contains two buttons: One to add it to the shopping cart and another one to close the window before proceeding. Add the product to the shopping cart that closes the window. Go back to the main browser and check to see whether the shopping cart was added.

This design will open a new window displaying the cross-sell/up-sell product. If desired, the design can fairly easy be changed to display the

cross-sell/up-sell product in the same window. The user can then use the browser button to go back to the parent product. To implement this change in behaviour we'll have to change the code shown in Figure 67. The new ShowSell function is shown in Figure 69.

```
<SCRIPT>
        function ShowSell(prrelprnbr) {
                self.location.href =
"/cgi-bin/ncommerce3/ProductDisplay?prmenbr=$(prmenbr)&prrfnbr="+prrelprnbr;
        }
</SCRIPT>
```

Figure 69. ShowSell function to display cross-sell/up-sell product in same window

Keep in mind, that cross-sell/up-sell product is now being displayed in the same window and frame, Net.Data "Sell" function in Figure 66 on page 113 should now be replaced with th code in Figure 70.

```
%{Cross sell, Up sell function starts here%}

%function(dtw_odbc) Sell(IN l_prtname){
  select plrelprnbr, prrfnbr, prnbr, prsdesc
  from prprrel, prreltype, product
  where
  plpub = 1 and
  prpub = 1 and
  prrfnbr = plrelprnbr and
  plprnbr = $(prrfnbr) and
  plprtnbr = prtrfnbr and
  prtname = '$(l_prtname)'

%REPORT{
   %ROW{
       @DTW_ASSIGN(prrelprnbr,$(V_prrfnbr))
     <tr>
         <td width=45% align=center><a
href="javascript:ShowSell($(prrelprnbr));"

onClick="ShowSell($(prrelprnbr));">$(V_prnbr)</a>   </td>
         <td width=55% align=left>$(V_prsdesc)</td>
     </tr>
   %}
  %}
  %MESSAGE{100:{ %} :continue %}
%}%{Cross sell, Up sell function ends here%}
```

Figure 70. Modified Cross-sell and up-sell functions

6.2.2 Import data through browser-based mass import tool

This section describes how to import data in the product-to-product relationship (PRPRREL) table. Before proceeding with this section, you must create the product relationship (PRREL) table and the product-to-product relationship (PRPRREL) table in the database. The creation of these two tables is explained in Section 6.2.1.1, “Creating product relationship tables in demomall database” on page 105.

The customization of the Net.Data macro to display the product page is also the same as explained in Section 6.2.1.3, “Customizing product display Net.Data macro” on page 111. A sample of the customized Net.Data macro, cat_product.d2w, is available in Appendix C.2, “Customized product display (cat_product.d2w) macro” on page 293.

Thus, this part would mainly concentrate on how to update the product-to-product relationship type (PRPRREL) table in the database from the merchant tool’s mass import utility available with the import catalog. We assume that merchants will be using the import catalog utility available in the merchant tool to import their product catalog. It is better to create the relationships between the product once when the product is being categorized. By doing so, related data is uploaded only once. To categorize the products that are to be sold online, an Excel template is available with the WCS SPE system. This template guides the creation of a mass import file for the product and its categories. This particular customization technique also extends the same excel template for creating data for the product relationship tables.

The steps involved in creating the mass import file and definition file and uploading them to the database can be described as follows:

1. Create the mass import file using a spreadsheet.
2. Append the definition file for new tables.
3. Upload data into database using import catalog option in merchant tool.

6.2.2.1 Creating a mass import file using a spread sheet

The Mass Import utility requires a file in ASCII delimited format. Some spreadsheet programs support this format; however, Lotus 1-2-3 does not. If you save a file as text in Lotus 1-2-3, it appears to be in tabular delimited format, but it is not. The characters separating columns are a series of spaces.

A template of the spread sheet file is available in the WCS SPE system. You can download the spreadsheet template file and use it to create your mass import file.

The template is available in the merchant tool of the WCS SPE system under the import catalog section. After creating the store, the merchant will wish to populate the store with products and information about them. To do this, he or she would upload the product catalog file, which contains the product information.

The merchant logs in to the merchant tool from the cspsite. He or she will click on the product catalog navigation bar on the import catalog submenu. It will display a window with an information button. If you click on the information button, it will display a window like that shown in Figure 71.

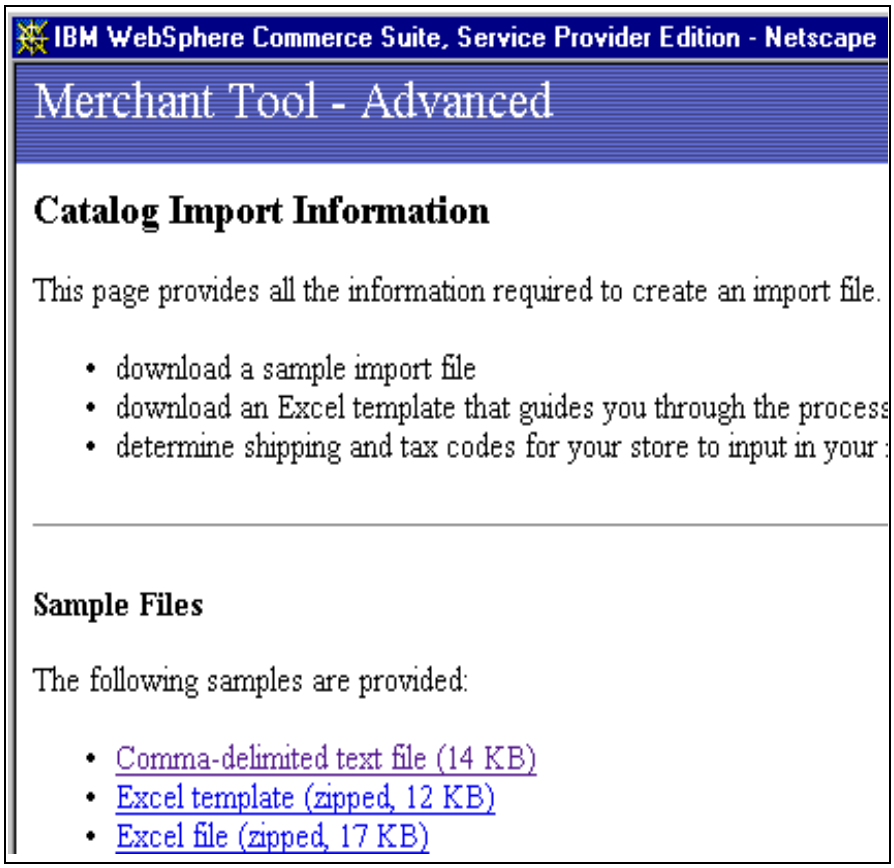


Figure 71. Merchant tool displaying product catalog features

This page provides all the information required to create an import file. You can:

- Download a sample import file (a comma-delimited text file)
- Download an Excel template that guides you through the process of creating an import file

Download the Excel template. You should customize the excel template to upload data for the new product-to-product relationship (PRPRREL) table. The technical details of the PRPRREL table are explained in Table 4 on page 106, and the details of the mass import file explained in Table 5 on page 110 will help create the new worksheet for creating product relationships in the sample.in work book. Save the file as a comma-separated file type (.csv). Open the file in the text editor to see the contents of the file in the comma-delimited format. A sample file, called sample.in, is available with the WCS SPE product to guide you in creating the mass import file.

As an example, Create a store with no products, and import the products and category to the store using the import catalog utility. Download the sample.in file from the merchant tool. Instead of using an excel template for creating a mass import file, Open the sample file in the text editor, and copy the following piece of data at the end of the file.

```
#PRPRREL,1,1,Compatible,,sku-d10,,sku-d1
#PRPRREL,1,1,Compatible,,sku-d10,,sku-d2
#PRPRREL,1,1,Cross-sell,,sku-d10,,sku-d3
#PRPRREL,1,1,Cross-sell,,sku-d10,,sku-d4
#PRPRREL,1,1,Cross-sell,,sku-d10,,sku-d5
#PRPRREL,1,1,Cross-sell,,sku-d10,,sku-d6
#PRPRREL,1,1,Up-sell,,sku-d10,,sku-d7
```

Save the file. Update the definition file as explained in the next section and import the file using the import catalog utility from the merchant tool.

6.2.2.2 Append the definition file for new tables

Before importing the data into the database from the comma-separated value format file, it is necessary to define the syntax of the new mass import file with the values of the new tables in a definition file. The definition file used for the mass import from the browser-based import catalog featured in the merchant tool is catimp.ini. This file is different from the default definition file of the mass import utility, massimpt.db2.in, which is available in the /usr/lpp/NetCommerce3/bin directory.

The catimp.ini file contains the definition of the tables to be uploaded except the definition of the new table we created. The next step is to append the file

with the definition of data for the new tables. This file is available in the /usr/lpp/NetCommerce3/Tools/public/catalog_import directory.

Open the catimp.ini file in a text editor and copy the following code at the end of the file:

```
#TABLE;prreltype
#COLUMNS;prtrfnbr;prtname;prtdesc;prtfield1;prtfield2;prtfield3
#UI;prtname
#REFNUM;prtrfnbr

#TABLE;prprrel
#COLUMNS;plrfnbr;plmenbr;plpub;plqty;plprtnbr;plsgnbr;plprnbr;plseqnbr;plr
elprnbr
#UI;plmenbr;plprnbr;plrelprnbr;plsgnbr;plprtnbr
#CONSTTOKEN;plmenbr;merchant
#REFNUM;plrfnbr
#FORREF;prprrel;plprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prprrel;plrelprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prprrel;plprtnbr;prreltype;prtrfnbr;prtname
```

Save the file. Now we have both the mass import file and the definition file in place for uploading.

6.2.2.3 Importing the file from the browser tool

To import data available in the mass import file, perform the following steps:

1. Log in to your merchant tool.
2. Select the **Product Catalog** menu in the navigation bar.
3. Click on the **Import Catalog** menu item.
4. This will display the window shown in Figure 72 on page 121.
5. Click on the **Browse** button to select your mass import file from your machine.
6. Select your file and open it.
7. Click on the **Import File** button to upload the data into the database.
8. The Merchant tool will send a confirmation stating that the catalog import has been submitted.
9. Close the merchant tool.

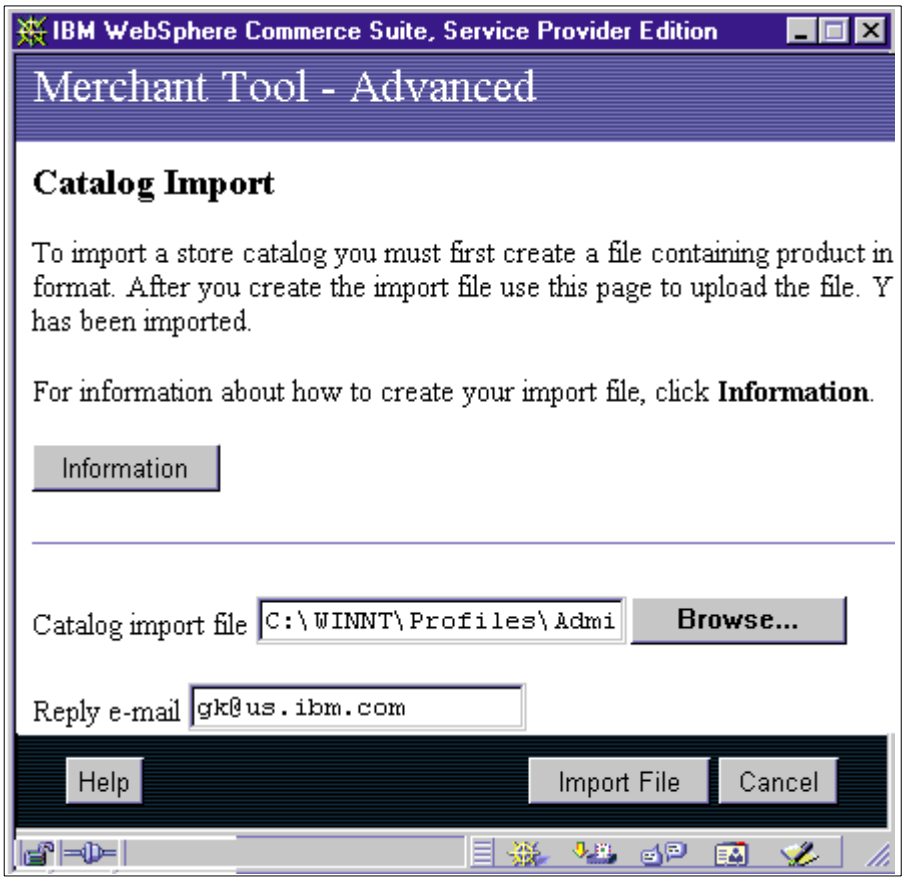


Figure 72. Import data using the catalog import utility

6.2.3 Creating product relationship by customizing merchant tool

This section describes the creation of a product relationship between products using a customized merchant tool. This technique is used to create relationships between products, and it can also be extended to remove relationships and modify the relationship between products. The scenario assumed is that the type of relationship existing between the products is as defined in Table 2 on page 103.

This technique also requires tables, such as the ones created in Section 6.2.1.1, “Creating product relationship tables in demomall database” on page 105. You should create the tables and populate the PRRELTYPE (product-to-product relationship type) table with the required data before proceeding. The customization of the Net.Data macro to display the product

page is also the same as explained in Section 6.2.1.3, “Customizing product display Net.Data macro” on page 111. A sample of customized Net.Data macro cat_product.d2w is available in Appendix C.2, “Customized product display (cat_product.d2w) macro” on page 293.

Thus, this part will mainly concentrate on how to update the PRPRREL (product-to-product relationship type) table in the database from the customized merchant tool instead of using a mass import utility. The mass import utility is the best option if the number of products is large, and, if the number of products is less, this method can be used to create a product relationship.

Since the merchant will be using the merchant tool, we have to add an extra feature to the tool’s menu that will link to our macro that will show the product relationship. On this page, the merchant can relate the product to another product belonging to the same store.

The macro is named productRel.d2w and is placed in the following directory:

```
/usr/lpp/NetCommerce3/macro/en_US/ncadmin/storemgr
```

The source code can be found in Appendix C, “Source code for cross and up selling” on page 291, macro file productRel.d2w**.**

To add this macro to the merchant tool for the advanced store under “Product Catalog”, do the following:

1. Locate and open the xml file, navigation.xml, for the navigation bar in the advanced store. At the time of this writing, it was found in:
   `/usr/lpp/NetCommerce3/Tools/xml/nchs/mtool/advanced`
2. In the file, locate the line that reads `<node name="productCatalog"`, and, just before the closing tag, `</node>`, add the following new link tag:

   ```
   <node name="RelateProduct"
   url="http://<hostname>/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/
   productRel.d2w/report?merfnbr=$env.merchant_id$"
   wholepage="true"
   image="/NCTools/images/clear.gif"
   users="siteAdmin storeAdmin catalogAdmin"/>
   ```

   where

   - **name** denotes the variable used to identify the link
   - **url** denotes the action of the link
   - **images** defines the image on the navigation bar to the left of the link
   - **users** defines which users are allowed to see this link

3. Save the XML file.
4. Locate and open the file, navigationNLS.properties, which is found in /usr/lpp/NetCommerce3/Tools/lib/nctools.jar. Unjar the file issuing the following from the command line:

   ```
   jar -xvf nctools.jar
   ```

5. This would extract into two sub folders called com and manifest. The navigationNLS. properties file can be found in the directory usr/lpp/NetCommerce3/Tools/lib/com/ibm/commerce/tools/nchs/properties
6. Open the navigationNLS.properties file using a text editor, and add the following line:

   ```
   productCatalog=Product Catalog
   ```

7. Save the file.
8. Re-create the nctools.jar file by issuing the following command from the command line

   ```
   jar -cf0 nctools.jar com
   ```

9. Restart WCS and Lotus Domino Go Web server to get the changes reflected.

After restarting WCS, you can launch the merchant tool to enable the feature for a particular store. Click on the **Relate Product** menu item in the navigation bar, and you will notice that the content frame displays the product relationship page, as shown in Figure 73 on page 124, from the macro, productRel.d2w.

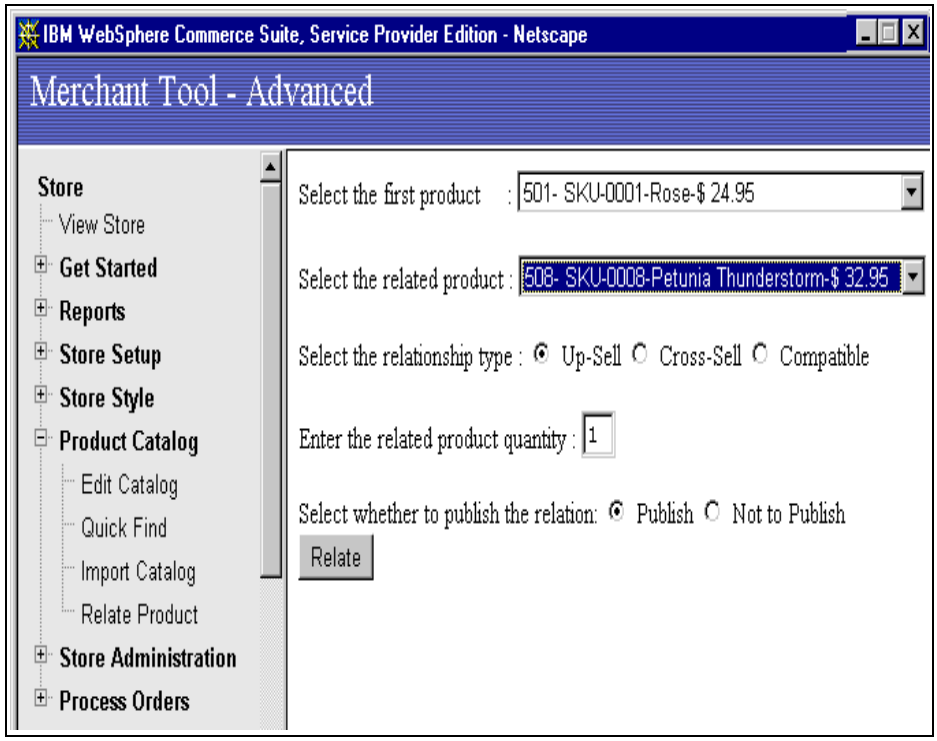


Figure 73. Customized merchant tool - product relationship data entry page

The merchant could select the product based on the details available in the list box. The list box shows details, such as the product number, item SKU number, product description, and the price of the product. These details are provided in the list box for selecting the product quickly based on the description and price. Then, the merchant could select the related product based on the reference number, item SKU, product description, and product price.

The merchant could provide the type of the relationship existing between the two products whether it is up-selling product, cross-selling product, or the related product is a compatible accessory to the main product. He is also required to give the quantity of the related product that could be sold together, such as two batteries per flash light. He or she may wish to publish the relation or not to publish the relation. Once he or she clicks on the Relate button, the product-to-product relationship (PRPRREL) table gets updated and the result is displayed in the confirmation page as shown in Figure 74 on page 125.

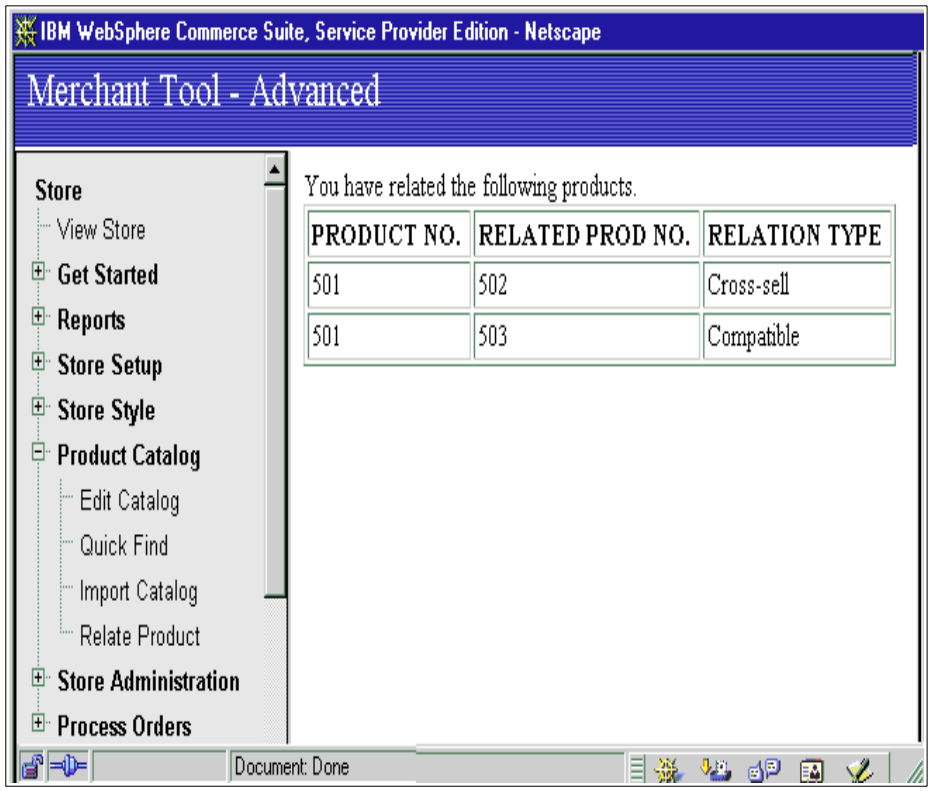


Figure 74. Customized merchant tool - Related products confirmation page

The confirmation page shows the details of the products related to each other and the type of the relationship between them as selected from the prprrel table.

Note

It is very important to note that Net.Data macro should not be used for updating the database in Net.Commerce system. We have done it here to make the interface simple. It is always recommended to use Commands and Overrideable Functions to achieve any update to the database in your implementation.

Once the product relationship is created, customize the Net.Data macro that displays the product page of your store by following the steps given in Section 6.2.1.3, “Customizing product display Net.Data macro” on page 111.

Chapter 7. External business system integration

This chapter describe how to integrate information from the WCS into the merchant's external business systems. The intended audience for this chapter may include the service providers and the merchants who own the online store and wish to integrate their legacy business system with a Net.Commerce system. In this chapter, we will mainly concentrate on how to integrate the existing order management system maintained by the merchants with the Net.Commerce system.

7.1 Integrating external business system with Net.Commerce

When we look at extending an e-commerce site to other external business systems, the first point of execution is the area of order management. Order management will mean the processes that are taking place after the shopper has placed the order in an online shopping store and given the valid payment and shipping details. This mainly deals with order fulfillment or customer fulfillment processes that are followed until the shopper receives the product or service for which he or she has paid. As orders are created in the commerce system, merchants want that order data to be passed to an existing accounting or order management application.

Connecting stores in WCS SPE to external business systems is a feasibility study with general implementation guidelines. Providing specific how to's is not reasonable because back end systems will vary from merchant to merchant. It is very important to note that the integration of merchant-related information to his or her order management or order fulfillment system should not crash the whole WCS SPE system. It means that the customization of an order processing system for an online store in Net.Commerce should be specific to that merchant-related functionality only. That is, customization for one merchant to his or her store level should not affect the other stores within the shared environment.

To achieve this functionality, a reference application, named Secure third party fulfillment [javafulfillment.exe, 422KB] can be downloaded from the net in the Webshpere commerce server reference site, which, at the time of this writing, was found in the following URL:

`www-4.ibm.com/software/webservers/commerce/community/process/refapps.html`

This application leverages the strength of Java to send an encrypted request for third-party fulfillment. Once an order is received from and confirmed to a shopper, this reference application helps you take that request and securely

send it outside your organization. You can download the application and unzip the package and the following the instructions as given in the product documentation, *Java Fulfillment: Integrating Java With IBM Net.Commerce*.

Chapter 8. Customizing the store creation wizard

This chapter will describe how to customize the store creation wizard for the advanced store. First, it will explain how to create new category and product pages for the merchant to choose from. Then, it will describe how to create new screens in the store creation wizard to allow the merchant to choose from these pages for their site. Lastly, it will describe how to alter the store creation process so that the new choices are implemented.
Using a step-by-step practical example, we hope to give a more detailed understanding of the store wizard and also show how an ISP can enhance functionality and offerings for their site.

8.1 The catalog screen

The catalog screen in the store creation wizard for the advanced store allows the merchant to select from predefined categories and products as they are creating their store. However, with the standard installation of WCS, these categories and products will only be displayed with the default pages as shown in Figure 75 below and Figure 76 on page 130.
Since not all merchants will be selling the same product range, it would benefit them immensely if they could choose from a number of different product and category pages. In the following sections, we will show how to create new product and category pages for the merchant to choose from and then how to customize the store creation wizard to offer this service.

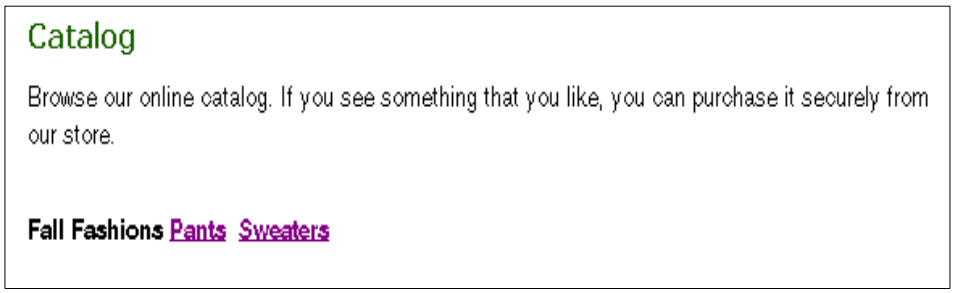


Figure 75. Default category page

Figure 76. Default product page

8.2 Creating custom category and product pages

First, we will look at how to create custom category and product pages for your site. We shall create two new category and product pages for the merchant to choose from. We will be using the ClassicStoreModel, but this will also hold for the ExpressStoreModel. The general rule for creating custom pages is to use the default macro as the base so that you are sure that they have all the necessary functions. Additional code should be kept in an include file, and any changes made to the original code should be as small as possible.

8.2.1 Custom category pages

The default category for the ClassicStoreModel is held in the file:

/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/cat_category.d2w

To create the new category pages, we took this file as the blueprint and created files under the same directory called cat_category2.d2w and cat_category3.d2w. The blueprint file contains the base functions required to create a category page. They are summarized as follows:

- **GET_CATEGORY_TITLEINFO()** - This stores the current category name and reference number in the variables, CGRYBANNERNAME and CGRYNUM.
- **DISPLAY_CATEGORIES()** - This displays all the subcategories with links that belong directly under the current category.
- **DISPLAY_PRODUCT_LIST_SINGLEPRICE()** - If the category displays products, it displays these in list form with single price information.
- **DISPLAY_PRODUCT_LIST_DUALPRICE()** - Similar to DISPLAY_PRODUCT_LIST_SINGLEPRICE() but with double price information in the situation of DEM and EURO.

All of these functions are essential and must be included to create a functioning category page.

8.2.1.1 Three level category example.

The first new category we created was based on adding a third-level category to the page when a merchant has a deep catalog tree. First, we took the original DISPLAY_CATEGORIES function, and, instead of displaying the result of the database query, we created a net.data Table variable to contain it. Figure 77 shows the changed function.

```
%function(dtw_odbc) DISPLAY_CATEGORIES(OUT table){

SELECT cgrfnbr, cgname, cgldesc
FROM   category, cgryrel
WHERE  crccgnbr=cgrfnbr and crpcgnbr=$(cgrfnbr) and crmenbr=$(cgmenbr) and cgpub=1
ORDER BY crseqnbr


    %MESSAGE{
        100:{ %} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
GET_CATEGORY_TITLEINFO()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}
```

Figure 77. DISPLAY_CATEGORIES function

Then, we created an include file containing the definitions and two new functions that we need to display the extra category information. First, a table variable. called catTable, is declared, which will hold the result from the DISPLAY_CATGEORIES function. The other definitions are standard net.data variables for the Table function:

```
%define{
catTable = %TABLE(20)
DTW_SET_TOTAL_ROWS="YES"
DTW_DEFAULT_REPORT = "NO"
%}
```

The DISPLAY_CATS macro function loops through the values in catTable and calls GET_CHILD_CATS for each, which displays any child categories found. The code for these functions is found in Figure 78 below and Figure 79 on page 133.

```
%function(dtw_odbc) GET_CHILD_CATS(IN newcgrfnbr){

SELECT cgrfnbr, cgname, cgldesc
FROM   category, cgryrel
WHERE  crccgnbr=cgrfnbr and crpcgnbr=$(newcgrfnbr) and crmenbr=$(cgmenbr) and
cgpub=1
ORDER BY crseqnbr

    %REPORT{
        <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
        <B>@DTW_TB_rGETV(catTable,loop,"2")</B>
        </FONT>
        </td><td>------</td><td>

        %ROW{
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
            <A
HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_cgrfnbr)&cg
menbr=$(MerchantRefNum)">
            <B>$(V_cgname)</B></A> 
            </FONT><br>
        %}
    %}
    %MESSAGE{
        100:{
        <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
        <A
HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(newcgrfnbr)&c
gmenbr=$(MerchantRefNum)">
        <B>@DTW_TB_rGETV(catTable,loop,"2")</B></A>
        </FONT> %} :continue
        default:{%}:continue
    %}
%}
```

Figure 78. GET_CHILD_CATS function

```
%MACRO_FUNCTION DISPLAY_CATS(){

@DTW_ASSIGN(loop,"1")
@DTW_TB_ROWS(catTable,totalrows)

%IF (totalrows != "0")
    <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
    <TR><TD align="left"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRY)</FONT></TD></TR>
    </TABLE></CENTER><BR>
    <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
    <TR><TD VALIGN="top" ><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(CGRYBANNERNAME)</B></FONT>
    </td><td>
    <table BORDER=0 CELLPADDING=4 CELLSPACING=0>

    %WHILE (loop <= totalrows) {
        <tr><td align=right>
        @DTW_TB_GETV(catTable,loop,"1",newcgrfnbr)
        @GET_CHILD_CATS(newcgrfnbr)
        </td></tr><tr><td><br></td></tr>
        @DTW_ADD(loop,"1",loop)
    %}

    </table>
    <BR></TD></TR><TR><TD><BR></TD></TR></TABLE></CENTER>
%ELSE
    @DTW_ASSIGN(NOCATEGORIESINCATEGORY, "YES")
    <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
    <TR><TD align="left"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRY)</FONT></TD></TR>
    </TABLE></CENTER><BR>
    <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
    <TR><TD VALIGN="top" ><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(CGRYBANNERNAME)</B></FONT>
    <BR></TD></TR><TR><TD><BR></TD></TR>
    </TABLE></CENTER>
%ENDIF
%}
```

Figure 79. DISPLAY_CATS function

The new functions were included in the cat_category2.d2w macro by adding the following statement in bold before the HTML_REPORT section of the macro:

```
%INCLUDE "more_cat.inc"
%{=====================================================%}
%{ HTML Report Section
%{=====================================================%}
%HTML_REPORT{
```

They are called just after the original function as in the following line:

```
<TR><TD>@DISPLAY_CATEGORIES(catTable)@DISPLAY_CATS()</TD></TR>
```

The resulting category page is shown in Figure 80, and the additional code is found in Appendix D.1, “Three level category example, file more_cat.inc” on page 301.

Figure 80. Three-level category custom page

8.2.1.2 The You are here custom category page

The second category page we designed contained additional information on the user’s location in a *You are here* bar. For example You are here >> Hand Tools >> Hammers. This is particularly useful if there is a deep category tree that allows the user to click back and forth along the various branches. Again, the cat_category.d2w macro was used as the base, and a new file, cat_category3.d2w, was created in the same directory. Two new functions were created in an include file as shown in Figure 81 on page 135. DISPLAY_YOU_ARE_HERE() is a macro function that starts at the current category and calls GET_CAT_PARENT repeatedly to get the previous categories until the root is found. These categories are stored using the JavaScript function, setupcategories, which adds an entry each time to the categories array. Once completed, the You_are_here string is displayed using the printyouarehere JavaScript function. The JavaScript functions are displayed in Figure 82 on page 136.

```
%define{
hasParent="TRUE"
globalref=""
%}

%function(dtw_odbc) GET_CAT_PARENT(IN cgrfnbr){
    SELECT cgname, cgrfnbr
    FROM category, cgryrel
    WHERE cgmenbr=$(MerchantRefNum)
        and crmenbr=$(MerchantRefNum)
        and CRCCGNBR=$(cgrfnbr)
        and cgrfnbr=CRPCGNBR
    %REPORT{
        @DTW_ASSIGN(hasParent,"TRUE")
        @DTW_ASSIGN(globalref,V_cgrfnbr)
        setupcategories('$(loop)','$(V_cgname)','$(V_cgrfnbr)')
    %}
    %MESSAGE{
        100:{ @DTW_ASSIGN(hasParent,"FALSE") %} :continue
        default:{@DTW_ASSIGN(hasParent,"FALSE")%} :continue
    %}
%}


%MACRO_FUNCTION DISPLAY_YOU_ARE_HERE(){

<script>
@DTW_ASSIGN(globalref,cgrfnbr)
@DTW_ASSIGN(loop,"0")

%WHILE (loop < "10" && hasParent == "TRUE"){
@GET_CAT_PARENT(globalref)
@DTW_ADD(loop,"1",loop)
%}

var cgmenbr = '$(MerchantRefNum)';
</script>

<tr><td><div align="left">
<font size="$(BodyFontSize)"  face="Arial, Helvetica, sans-serif">
 <img src="/EXT_pages/images/dot_336699.gif" width="7" height="7"> You are here:
<a href="javascript:top.location.reload();">Home</a>
<font size="$(BodyFontSize)" face="Arial, Helvetica, sans-serif">
@DTW_SUBTRACT(loop,"2",loop)

<script>
printyouarehere('$(loop)')
</script>

&#187;$(CGRYBANNERNAME)</font></div></td></tr>
%}
```

Figure 81. YOU_ARE_HERE_CAT include file

```
var categories = new Array();
var list = "";

function CreateCats(name,ref ) {
this.name=name;
this.ref=ref;
return this;}

function setupcategories(i,name,ref){
categories[i]=new CreateCats(name,ref );}

function printyouarehere(total){
var index=total;
while( categories[index] && index > -1){
printcategory(index);
index--;
}
document.write(list);}

function printcategory(index){
list += " &#187; ";
list += "<a href=/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=";
list += categories[index].ref+"&cgmenbr="+cgmenbr+">";
list += categories[index].name+"</a>";
}
```

Figure 82. YOU_ARE_HERE JavaScript functions

Only a few changes were then made to the original cat_category.d2w file. The files are included by adding the bold lines of code shown in Figure 83.

```
%{=================================================%}
%{ Extension section
%{=================================================%}

%INCLUDE "you_are_here_cat.inc"


%{=================================================%}
%{ HTML Report Section
%{=================================================%}

%HTML_REPORT{

<HTML>
@GET_CATEGORY_TITLEINFO()
<HEAD>
    <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1999 01:01:01 GMT">
    <SCRIPT SRC="/EXT_pages/javascript/you_are_here.js"></SCRIPT>
```

Figure 83. Include statements for cat_category3.d2w

The DISPLAY_YOU_ARE_HERE function is called just before the existing DISPLAY_CATEGORIES function.

```
<TR><TD>@DISPLAY_YOU_ARE_HERE()</TD></TR>
<TR><TD>@DISPLAY_CATEGORIES()</TD></TR>
```

The macro results in the screen shown in Figure 84, and the code is found in Appendix D.2, “You are here custom page” on page 302.

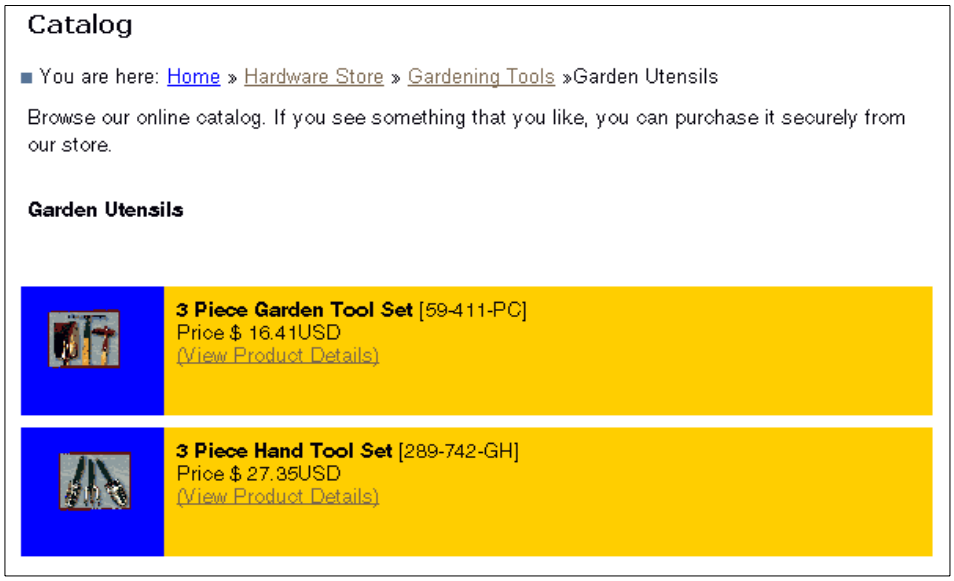


Figure 84. YOU_ARE_HERE custom category page

8.2.2 Custom product pages

The default product page for the ClassicStoreModel is held in the file:

```
/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/cat_product.d2w
```

Again, as in the category example, it is very important to use this file as a base for your new custom product pages. Especially since this macro file is a little more complicated than the previous example. The base functions are as follows:

- **GET_PRODUCTNAME()** - This stores the product name in the PRODUCTNAME variable.
- **GET_DISTINCTATTRIBUTES()** - This stores the number of attributes that a product has in the FLAG_ATTRIBUTES variable.

- **DISPLAY_PRODUCT_IMAGE()** - This displays the product large image.
- **DISPLAY_PRODUCT_INFO()** - This displays the product SKU number and product short and long description.
- **DISPLAY_PRODATTR()**- This displays each of the product attributes in a drop down list with their values. It also displays the *Add to Shopping Cart* button.
- **DISPLAY_PRODUCT_SINGLEPRICE()** - This displays single product price.
- **DISPLAY_PRODUCT_DUALPRICE()** - This displays double product price when the user has selected two currencies.
- **DISPLAY_PRODUCT_SINGLEPRICE_RANGE()** - The product may be on special offer; so, this function displays the previous prices along with the current special-offer price.
- **DISPLAY_PRODUCT_DUALPRICE_RANGE()** - This is as above but only for dual currencies.

8.2.2.1 Shortened product page example

In the first example, we took the cat_product.d2w macro as a base and created a new file, called cat_product2.d2, under the same directory. We are aiming to create a new product page that will take up less space on the main frame so the user won't have to scroll. The principal change we made was to move the DISPLAY_ITEMS_DROPDOWN() function so that it comes before the DISPLAY_PRODUCT_SINGLEPRICE_RANGE() and DISPLAY_PRODUCT_DUALPRICE_RANGE() functions and to take the *Add to Shopping Cart* button out of the items function so that the items and prices could be displayed in the same HTML table. Part of the code is shown in Figure 85 on page 138.

```
    %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor == NULL || ConvDivideFactor ==
NULL))
        @DISPLAY_ITEMS_DROPDOWN()
        @DISPLAY_PRODUCT_SINGLEPRICE_RANGE()
    %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor != NULL && ConvDivideFactor !=
NULL))
        @DISPLAY_ITEMS_DROPDOWN()
        @DISPLAY_PRODUCT_DUALPRICE_RANGE()
    %ENDIF
    %IF (SHOWADDTOCART == "YES")
        <p><INPUT TYPE="hidden" NAME="comment" VALUE="$(LBL_COMMENTS): ">
        <INPUT TYPE=SUBMIT VALUE="@DTW_rUPPERCASE(BUT_ADDTOSHOPCART)">
        </TD></TR></TABLE>
    %ENDIF
```

Figure 85. Shortened product page example code

The remaining changes were to the HTML table tags and the space tags to shorten the page. Figure 86 shows how it appears, and the final code is in Appendix D.3, “Shortened product page example, cat_product2.d2w file” on page 304.

.

Figure 86. Shortened custom product page

8.2.2.2 “You are here” custom product page.

The second product page we designed was an extension on the previous “You are here” category example. This time the product name would be included and the category history would begin from a product.
Again the cat_product.d2w macro was used as the base and a file cat_product3.d2w created in the same directory. But we also used our previous code you_are_here.js and you_are_here_cat.inc, which we renamed you_are_here.inc. One extra function was added to you_are_here.inc, that retrieves the starting parent category of the product as shown in Figure 87.

.

```
%function(dtw_odbc) GET_PROD_CAT(IN prrfnbr){
    SELECT cgname, cgrfnbr
    FROM category, cgprrel
    WHERE cgmenbr=$(MerchantRefNum)
        and cpmenbr=$(MerchantRefNum)
        and cpprnbr=$(prrfnbr)
        and cgrfnbr=cpcgnbr
    %REPORT{
        @DTW_ASSIGN(hasParent,"TRUE")
        @DTW_ASSIGN(globalref,V_cgrfnbr)
        setupcategories(0,'$(V_cgname)','$(V_cgrfnbr)')
    %}
    %MESSAGE{
        100:{ @DTW_ASSIGN(hasParent,"FALSE") %} :continue
        default:{@DTW_ASSIGN(hasParent,"FALSE")%}:continue
    %}
%}
```

Figure 87. GET_PROD_CAT function

Also, the function, DISPLAY_YOU_ARE_HERE, needed to be changed to include the product name. The changes are shown in bold in Figure 88 on page 140.

```
%MACRO_FUNCTION DISPLAY_YOU_ARE_HERE(){

<script>
@GET_PROD_CAT(prrfnbr)
@DTW_ASSIGN(loop,"1")
%WHILE (loop < "10" && hasParent == "TRUE"){
@GET_CAT_PARENT(globalref)
@DTW_ADD(loop,"1",loop)
%}
var cgmenbr = '$(MerchantRefNum)';
</script>

<tr bgcolor="#FFFFFF"><td><div align="left">
<font size="$(BodyFontSize)"  face="Arial, Helvetica, sans-serif"> <img
src="/EXT_pages/images/dot_336699.gif" width="7" height="7"> You are here: <a
href="javascript:top.location.reload();">Home</a>
<font size="$(BodyFontSize)" face="Arial, Helvetica, sans-serif">
@DTW_SUBTRACT(loop,"2",loop)

<script>
printyouarehere('$(loop)')
</script>
&#187;$(PRODUCTNAME)
</font></div></td></tr>
%}
```

Figure 88. DISPLAY_YOU_ARE_HERE function for custom product page

The Java script file remained the same and they are included and called as they are for the You_are_here category example. The custom code created is found in Appendix D.4, “The “You are here” custom product page and file you_are_here.inc” on page 318, and the resultant page is shown in Figure 89 on page 141.

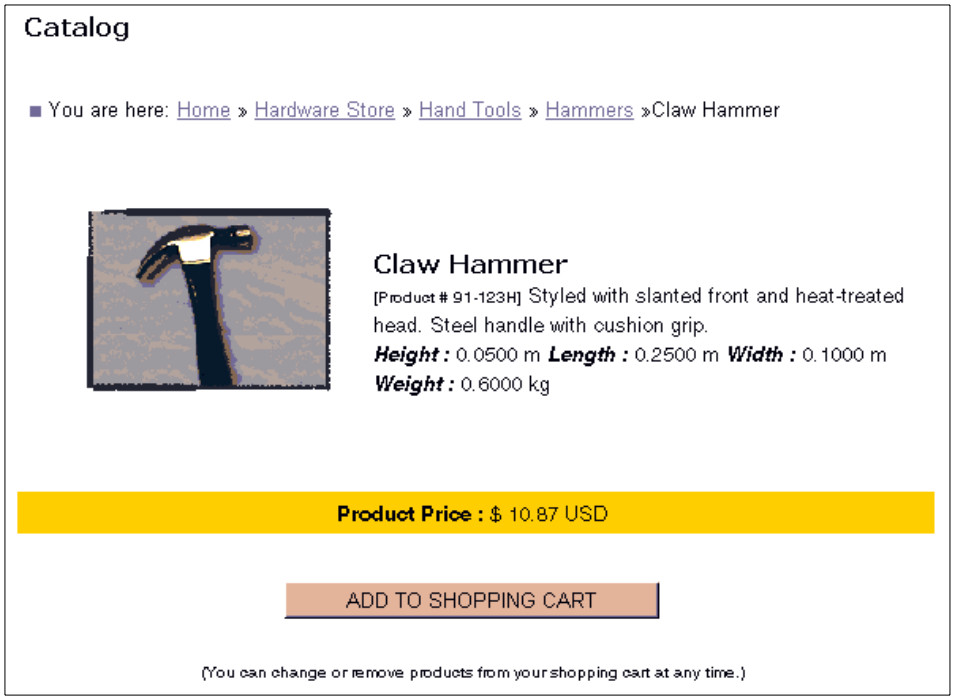


Figure 89. The “YOU_ARE_HERE custom product page

8.2.3 How the store creation wizard works

This sections explains in detail how the wizard operates and then describes the code we created to add the new panels. For full implementation details, see Appendix D, “Store creation wizard sample code” on page 301. The store creation wizard contains a number of panels that offer, for example, alternative themes or banner bars from which the merchant chooses to determine how his or her store appears. These choices are stored by the wizard and used to create the merchant’s store when they press the Finish button at the end of the creation process. This information is held as an object-called model in the store creation wizard’s client browser. Each panel adds more values to this object. When Finish is pressed on the final screen, a Java script function, convertToXML(), is called, which converts this object into a number of XML parameters that are then sent to the servlet,

MerchantAdmin. This servlet then does all the work of creating the new store based on the values in the XML parameters. In regards to the category and product pages, this servlet carries out the following actions:

1. It creates the new store files for the category and product pages based on an XML tag specified in an XML parameter sent from the final screen.
2. It places these new files in the new store's home macro directory. This is:

```
/usr/lpp/NetCommerce3/macro/en_US/category/<merchant reference number>/

/usr/lpp/NetCommerce3/macro/en_US/product/<merchant reference number>/
```

You can obtain the merchant reference number by running the sql shown in Figure 90 where mename equals the merchant's name.

```
-------------------------- Command entered ----------------------------
select merfnbr from merchant where mename='bellavista'
-----------------------------------------------------------------------

MERFNBR
-----------
       2147

  1 record(s) selected.
```

Figure 90. SQL to select merchant's reference number

3. Inserts new entries into the macros table for this new merchant. These new entries specify which macro the product displays and the category display tasks use. Figure 91 shows the sql to display these values.

```
-------------------------- Command entered ----------------------------
select tkrfnbr, tkname, mamernbr, mafilename from macros, tasks where
tkrfnbr=marfnbr and mamernbr = 2147 and (tkrfnbr=9042 or tkrfnbr=9043)
-----------------------------------------------------------------------

TKRFNBR   TKNAME          MAMERNBR   MAFILENAME
-------------------------------------------------------------------
9043  DEFAULT_CAT_TEMPLATE    2147 2147/cat_category.d2w
9042  DEFAULT_PROD_TEMPLATE    2147 2147/cat_product.d2w

  2 record(s) selected.
```

Figure 91. SQL to select product and category macros

In the example in Figure 91, we can see that the task, DEFAULT_CAT_TEMPLATE, uses the template in the directory:

```
/usr/lpp/NetCommerce3/macro/en_US/category/2147/cat_category.d2w
```

where the root directory for category macros is:

```
/usr/lpp/NetCommerce3/macro/en_US/category/
```

Here, we have the contents of this template, which we see is based on the ClassicStoreModel's cat_category.d2w macro:

```
%include "2147\include.inc"
%include "2147\theme.inc"
%include "ClassicStoreModel\cat_category.d2w"
```

Which files to create and use as a template are defined in the XML parameters sent in the final screen. Therefore, the key to altering the category and product files created for a merchant lies in changing the content of the final XML parameter. Figure 92 shows the section of the XML that contains the definitions for the category and product macros.

```
 <task>
     <refno>9043</refno>
     <file>cat_category.d2w</file>
  </task>
   <task>
     <refno>9042</refno>
     <file>cat_product.d2w</file>
  </task>

.....

   <macro>
     <baseDir>categoryBase</baseDir>
     <name>cat_category</name>
     <generator>cat_category</generator>
  </macro>
   <macro>
     <baseDir>macroBase</baseDir>
     <name>cat_category</name>
     <generator>cat_category</generator>
  </macro>
   <macro>
     <baseDir>productBase</baseDir>
     <name>cat_product</name>
     <generator>cat_product</generator>
  </macro>
   <macro>
     <baseDir>macroBase</baseDir>
     <name>cat_product</name>
     <generator>cat_product</generator>
  </macro>
```

Figure 92. XML definitions for category and product pages

Appendix D.5, “Full macros XML parameter” on page 319, shows the full macros XML parameter.

The <task> tag represents the task reference number (tasks.tkrfnbr) and macro file (macros.mafilename) to be inserted into the macros table for that task. The <macro> tags define the base product or category macro that should be used when creating the new file. It has a number of elements. <baseDir> is to define the directory for this macro. For example, we have the value categoryBase, which translates as the following directory:

```
/usr/lpp/NetCommerce3/macro/en_US/category/
```

The <name> tag is the name of the new file, and the <generator> tag is used to define which macro it is to be generated from; so, if we wanted to use the cat_category3.d2w macro, which we just created in our new store, the resultant XML tags should appear as shown in Figure 93.

```
<task>
    <refno>9043</refno>
    <file>cat_category3.d2w</file>
 </task>
  <macro>
    <baseDir>categoryBase</baseDir>
    <name>cat_category3</name>
    <generator>cat_category3</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_category3</name>
    <generator>cat_category3</generator>
 </macro>
```

Figure 93. Altered XML parameter for custom category page

The MerchantAdmin servlet would then create the entries for the new file, cat_category3.d2w, instead of the default.

8.2.3.1 Changing the macro XML settings

There are a number of alternative solutions for changing the XML settings. It would be good to take advantage of WebSphere’s XML parser, write a servlet that would alter the macros XML, and then use filtering to pass the changed parameters to the MerchantAdmin servlet. Unfortunately, in practice, this has proven too slow and cumbersome. The idea we chose was to write a JavaScript function that would change the XML parameter once it had been created by the JavaScript function, convertToXML(). This would save any unneccessary parameter requests to the server and would allow all the processing to take place on the client side.

To represent the elements that we want to change in the XML parameter, we created a JavaScript array variable, called newElements, which is defined as:

```
newElements.orig
newElements.newvalue
```

Where `orig` contains the default XML tag and `newvalue` contains the changed version. Then, as shown in Figure 94, we wrote the JavaScript function, changeXML, which parses through the XML parameter one line at a time searching for a match to *orig.* If found, it changes the string to newvalue.

```
function nospaces(tempstring){
var tempstring2="";
for ( j=0; j < tempstring.length; j++)
    {if (tempstring.charAt(j) != " ")
        tempstring2 += tempstring.charAt(j);
    }
return tempstring2;}

function changeXML(xmlstring) {
nlcode = "10";
var tempstring="";
var tempstring2="";
var newxmlstring="";
var index = 0;
var b = false;

for ( i=0; i < xmlstring.length; i++){
if (xmlstring.charCodeAt(i) == nlcode ){
    // go around all elements and check for a hit. If yes then change it.
    index = 0;
    b = false;
    tempstring2 = nospaces(tempstring);
    while( newElements[index] && !b ) {
        if ( newElements[index].orig == tempstring2 ){
            tempstring = newElements[index].newvalue;
            b = true;
            break;
        } // if
        index++;
    } //while
    newxmlstring += tempstring + xmlstring.charAt(i);
    tempstring="";
    }
else
    tempstring += xmlstring.charAt(i);}
return newxmlstring;
}
```

Figure 94. ChangeXML JavaScript function

But, we also need to create values for this newElements array. The JavaScript function, setXMLChanges, shown in Figure 95, is used in the category and

product selection panels in the store creation wizard. When the merchant selects a category page, setXMLchanges is called to create or update the contents of the newElements array.

```
// arrays for changed Elements
var newElements = new Array();

function CreateElement( orig, newvalue ) {
this.orig = orig
this.newvalue=newvalue;
return this;
}

function UpdateElement( orig, newvalue, index ) {
newElements[index].orig=orig;
newElements[index].newvalue=newvalue;
return this;
}

function getarrayindex_Elements( orig ) {
var newindex=newElements.length;
var b = true;
var index=0;
while( newElements[index] && b){
if (newElements[index].orig == orig)
    {b=false;newindex=index;break;}
index++;
}
return newindex;
}

function setXMLChanges(orig_page, chosen_page, orig_base , chosen_base, orig_gen,
chosen_gen){
var i=getarrayindex_Elements( orig_page );
if (i==newElements.length)
newElements[i]=new CreateElement(orig_page,chosen_page);
else
UpdateElement(orig_page,chosen_page,i);

i=getarrayindex_Elements( orig_base );
if (i==newElements.length)
newElements[i]=new CreateElement(orig_base,chosen_base);
else
UpdateElement(orig_base,chosen_base,i);

i=getarrayindex_Elements( orig_gen );
if (i==newElements.length)
newElements[i]=new CreateElement(orig_gen,chosen_gen);
else
UpdateElement(orig_gen,chosen_gen,i);
}
```

Figure 95. SetXMLchanges JavaScript function

All of the above JavaScript is contained in a single file, called CheckForChanges.js. See Appendix D.6, “CheckForChanges.js JavaScript” on page 324. It is included in the wizard using its model template file, which defines the top frame of the wizard window. Here is the file:

```
/usr/lpp/NetCommerce3/Tools/mpg_templates/common/Model.tem
```

It was included just above the other JavaScript files in the following way:

```
<SCRIPT SRC="/EXT_NCTools/javascript/CheckForChanges.js"></SCRIPT>
<SCRIPT SRC="/NCTools/javascript/ConvertToXML.js"></SCRIPT>
<SCRIPT SRC="/NCTools/javascript/NumberFormat.js"></SCRIPT>
```

To change the XML just created by calling the changeXML() fn, we altered the verify panel for the advanced store. This is found in:

/usr/lpp/NetCommerce3/Tools/mpg_templates/nchs/store_creator/advanced/Verify.tem

And, our new function was called after the XML[1] parameter was created. The XML[1] parameter contains the macro information. You will see the new code in bold below:

```
document.f1.XML[0].value = top.convertToXML(pageObject    , "pageXML");
document.f1.XML[1].value = top.convertToXML(macros        , "macros");
document.f1.XML[1].value = top.changeXML(document.f1.XML[1].value);
document.f1.XML[2].value = top.convertToXML(layouts       , "layoutXML");
```

How we called the setXMLchanges function will be shown in the next section where we will create the new panels for the store creation wizard.

8.2.4 Creating new category and products panels

The next step is to create the new screens or panels in the store creation wizard that will allow the merchant to choose from our category and product pages.
Even though the templates in WCS are created in MPG, that does not mean that new ones must also be created in MPG. This gives us the freedom to choose whatever Internet technology we prefer and is supported by WCS from simple HTML or net.data macro. For our example, we have chosen to use servlets, JSPs and XML files. This will introduce the new underlying WebSphere technology that is introduced in this latest release of WCS. It also follows along the same methodology as used in MPG to separate the processing logic from the display and data logic. The XML is used for representing the data, the JSP for display, and the servlet and bean classes for functionality. Our design is as shown in Figure 96 on page 148.

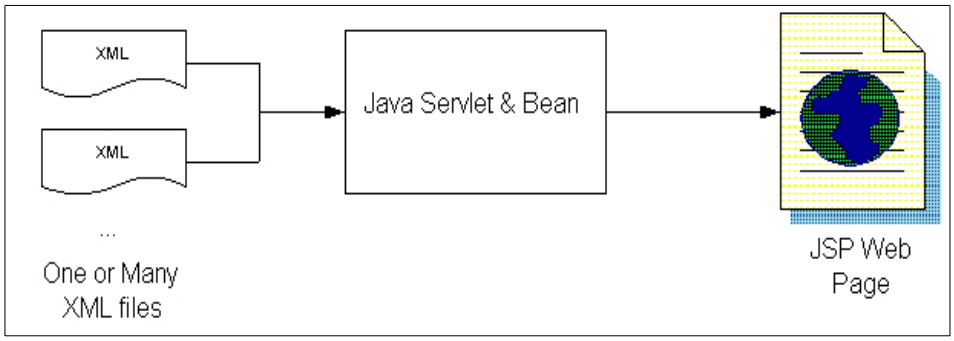


Figure 96. Design flow for new wizard panels

In this example, we want to create a generic servlet that could be used to create any number of new panels in the store creation wizard. These panels will allow the merchant to choose from a variety of pages for the one function, such as different login screens or order summary pages. The only change would be in the XML file and, maybe, but not necessarily, the JSP file. There is an HTML version of the panels in Appendix D.10, "HTML panel pages and file Displaycats./" on page 331 which has hard coded the category and product choices, if you require a simpler solution.

8.2.4.1 XML files

We will use XML files to contain the information about the various product or category pages we want to show to the merchant. Since we also plan to use only one display JSP page, we will use the XML to contain display information, such as the image for each page and the title. We created two XML files, category.xml and product.xml, under the /appldata/XML/ directory according to our own format as shown in Figure 97 and Figure 98 on page 149.

```
<?xml version="1.0"?>
<newmacros>
<title value="Please choose from the category pages below:"/>
<origpage value="cat_category.d2w"/>
<origbase value="cat_category"/>
<newtask file="cat_category.d2w" base="cat_category" image="cat1.gif" desc="Default
page"/>
<newtask file="cat_category2.d2w" base="cat_category2" image="cat2.gif" desc="More
Categories"/>
<newtask file="cat_category3.d2w" base="cat_category3" image="cat3.gif"
desc="You_Are_Here Info"/>
</newmacros>
```

Figure 97. Category.xml

```
<?xml version="1.0"?>
<newmacros>
<title value="Please choose from the product pages below:"/>
<origpage value="cat_product.d2w"/>
<origbase value="cat_product"/>
<newtask file="cat_product.d2w" base="cat_product" image="prod1.gif" desc="Default
page"/>
<newtask file="cat_product2.d2w" base="cat_product2" image="prod2.gif" desc="Shortened
page"/>
<newtask file="cat_product3.d2w" base="cat_product3" image="prod3.gif"
desc="You_Are_Here Info"/>
</newmacros>
```

Figure 98. Product.xml

Description of the tags:

- <title> - This is the title to display on the panel page.
- <origpage> - This is the default macro value.
- <origbase> - This is the default base value.
- <newtask> -This describes each of the choices to be presented to the merchant.
- <file> - This is the new macro filename.
- <base> - This is the new base value.
- <image> - This is the image to appear to represent this page choice.
- <desc> - This is the description text to display for each page.

8.2.4.2 The Java servlet

The Java servlet class will parse the XML file specified in the parameters and set the values of the macrosBean (see Section 8.2.4.3, “macrosBean” on page 153) according to the content of the XML file. Then, it will call the JSP page, specified as a request parameter, with the macrosBean included in the response.

The parameters sent to the servlet are:

xmlfile=product.xml
jspfile=/EXT_pages/html/DisplayPages.jsp

We also create an initial parameter for the servlet as:

xmlpath=/appldata/XML/

Appendix D.13.2, “Websphere configuration” on page 336, shows how to create an initial parameter. Therefore, the xmls are to be found in XMLPATH + XMLFILE.

Figure 99, shows the imports required for the servlet. It is important to have these classes in your class path when compiling. See Appendix D.13.2, “Websphere configuration” on page 336.

```
/*********************************************************************
* DisplayPagesServlet
*********************************************************************/

// These imports are for the XML parsing
import org.w3c.dom.*;
import com.ibm.xml.parsers.*;
import com.ibm.xml.parser.*;

import java.io.*;
import java.beans.Beans;

// These are for the servlet methods
import javax.servlet.*;
import javax.servlet.http.*;

//We import our custom created Bean
import macrosBean;
```

Figure 99. Imports for DisplayPagesServlet

Figure 100 on page 151 shows the parseXML method. We are using the IBM XML parser for Java (XML4J). This is pure Java software that can read XML data and generate its structured tree. This then allows the elements of the tree to be accessed as objects using DOM (Document Object Model). First, we obtain the root of the tree:

Element root = (Element)doc.getDocumentElement()

And we then use this as the reference to obtain the other elements. In our example, we know the names of the elements we are looking for but not their attributes. The following statement retrieves the attribute value for element title:

((Element) root.getElementsByTagName("title").item(0)).getAttribute("value")

We then call assign functions in the macrosBean class (see Section 8.2.4.3, “macrosBean” on page 153) to store these tag values.

mBean.settitle(((Element)
root.getElementsByTagName("title").item(0)).getAttribute("value"))

For further information concerning WebSphere and XML, refer to the redbook, The XML Files: *Using XML and XSL with IBM WebSphere 3.0*, SG24-5479.

```
 public void parseXML(String xmlpath, macrosBean mBean)
      throws ServletException, IOException
  {

    try{
    // declare a DOMparser on our xml file.
    DOMParser parser = new DOMParser();
    parser.parse(xmlpath);
    // Then we retreive the document root of the file.
    Document doc = null;
    doc = parser.getDocument();
    Element root = (Element)doc.getDocumentElement();

    // Having retrieved the root we use that to retreive the other elements by tag name.
    // which we then use as parameters in assign calls to the macroBean
    mBean.settitle(((Element)
root.getElementsByTagName("title").item(0)).getAttribute("value"));
    mBean.setorigpage(((Element)
root.getElementsByTagName("origpage").item(0)).getAttribute("value"));
    mBean.setorigbase(((Element)
root.getElementsByTagName("origbase").item(0)).getAttribute("value"));

    //Here we get the list of elements containing the newtask tag
    NodeList elements = root.getElementsByTagName("newtask");

    //Then we loop around this list to obtain the other tag values
    for (int i = 0; i < elements.getLength(); i++) {
    Element orderItem = (Element)elements.item(i);
    mBean.setfile(((Element)elements.item(i)).getAttribute("file"),i);
    mBean.setbase(((Element)elements.item(i)).getAttribute("base"),i);
    mBean.setimage(((Element)elements.item(i)).getAttribute("image"),i);
    mBean.setdesc(((Element)elements.item(i)).getAttribute("desc"),i);
    }

    }
    catch(FileNotFoundException e1){
    e1.printStackTrace();
    }
    catch(Exception e2){
    e2.printStackTrace();
    }
} //end of parseXML method
```

Figure 100. ParseXML method

Figure 101 on page 153 shows the main body of the servlet. The doGet and doPost methods both call the performTask method. The full code is found in Appendix D.7, “DisplayPagesServlet.java servlet” on page 326.

```
public class DisplayPagesServlet extends HttpServlet
{

   public void doGet(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
  {
        performTask(req, res);
    }

   public void doPost(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
  {
        performTask(req, res);
    }

   public void performTask(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
  {

     // Get parameters both init and from request.
    String xmlValue = getServletConfig().getInitParameter("xmlpath") +
req.getParameter("xmlfile");
    String jspValue = req.getParameter("jspfile");

      macrosBean mBean;

    // create an instance of the macrosbean
      try
     {
         mBean = (macrosBean) Beans.instantiate(this.getClass().getClassLoader(),
"macrosBean");
      }
      catch (Exception ex)
      {
         throw new ServletException("Can't create BEAN of class macrosBean: "
                                    + ex.getMessage());
      }

      // Call the XML parse function
   parseXML(xmlValue,mBean);

      // To send the Bean to a JSP file for content formatting and display
      // 1) Set the Bean as an attribute in the current request object
      ((com.sun.server.http.HttpServiceRequest) req).setAttribute("mBean", mBean);

      // 2) Use callPage to invoke the JSP file and pass the current request object
      ((com.sun.server.http.HttpServiceResponse) res).callPage(jspValue, req);

   } // end of performTask method
```

Figure 101. DisplayPagesServlet

8.2.4.3 macrosBean

For storing the data for access by the JSP page we have created a bean class called macrosBean. The XML file is parsed and the values in the bean set by

the servlet discussed in Section 8.2.4.2, “The Java servlet” on page 149. The macrosBean contains the variables and methods for storing and retrieving this data. Figure 102 shows the beginning of the macrosBean class and two methods to set and retrieve the title value. The full code is found in Appendix D.8, “MacrosBean.java class” on page 328. The java code in the files MacrosBean.java and DisplayPagesServlet.java is compiled using the following commands

Compiling MacrosBean.java

javac -classpath .;D:\IBM\JDK\lib\classes.zip;D:\IBM\WAServer\lib\jsdk.jar MacrosBean.java

Compiling DisplayPagesServlet.java:

javac -classpath .;D:\IBM\JDK\lib\classes.zip;D:\IBM\WAServer\lib\jsdk.jar; D:\IBM\WAServer\lib\jst.jar;D:\IBM\NetCommerce3\Tools\lib\xml4j.jar DisplayPagesServlet.java

```
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;

public class macrosBean extends HttpServlet {
private String title = "";
private String origpage = "";
private String origbase = "";
private static final int numpages = 3;
private String[] file = new String[numpages];
private String[] base = new String[numpages];
private String[] image = new String[numpages];
private String[] desc = new String[numpages];

public void service(HttpServletRequest req, HttpServletResponse res)
throws IOException
{
}
public void settitle(String value) {
this.title = value;
}
public String gettitle() {
return title;
}

....
```

Figure 102. MacrosBean class

8.2.4.4 JSP page

We created a generic JSP page, DisplayPages.jsp, to display the new page choices to the merchant. First, it accesses the macrosBean created by the DisplayPagesServlet:

<bean name="mBean" type="macrosBean" introspect="no" create="no" scope="request"></bean>

Then, it retrieves the default page values from the bean by calling the methods directly and then stores them as Java script variables.

```
//getting the variables from the bean.
<% out.println("var orig_page='<file>" + mBean.getorigpage() +"</file>';"
);%>
<% out.println("var orig_basetemp='" + mBean.getorigbase() +"';");%>
```

The various page choices, macro file and base file, are stored in the following JavaScript array variable:

```
theOptions.file
theOptions.baseDir
```

Figure 103 shows the code for doing this.

```
//theOptions variable holds the different page choices for the user
var theOptions = new Array();
var list = "";

function CreateOptions(file, baseDir ) {
this.file=file;
this.baseDir=baseDir;
return this;
}
//This function prints panel choices and sets it's values in variable theOptions.
function setuppage(filevalue,basevalue,j,image,desc)
{
var listbase="<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>";
var listbase2 = "<td><p></td><td><input type=radio name=choice value=" +j + "
onclick=storesettings(this.value)></td>";
var listimage = "<img src=/EXT_pages/images/" + image+" width=130 height=150>"
list += listbase2 + listbase + listimage +"</td></tr><tr><td><font face=Arial,
Helvetica, sans-serif>"+ desc +"</font></td></tr></table></td>";
theOptions[j]=new CreateOptions(filevalue,basevalue);
}
//prints out all the choices on the panel
function printoptions(){
document.write(list);
}
```

Figure 103. Storing the pages choices in the JSP page.

The setuppage() function is called for each choice retrieved from the bean. It adds this choice to the theOptions array and also creates the HTML tags to define how it is to be displayed in the variable list. Then, the printoptions() function is called to display all the choices on the panel. The code in Figure 104 on page 156 uses the JSP, REPEAT tag to loop around the choices in the bean and call the setuppage() function. Finally, the printoptions() function is called.

```
<script>
<REPEAT INDEX="i">

<% out.println("setuppage('" + mBean.getfile(i)+"','"+
mBean.getbase(i)+"',"+i+",'"+mBean.getimage(i)+"','"+mBean.getdesc(i)+"')" );%>

</REPEAT>
printoptions();
</script>
```

Figure 104. Displaying the page choices on the panel

Full implementation details and code can be found in Appendix D.9, “DisplayPages.jsp page” on page 330. How the new panels appear can be seen in Figure 107 on page 159 and Figure 108 on page 160.

8.2.5 Adding new product and category panels

Now, we should have everything complete and ready to integrate into the store creation wizard. The following describes the step-by-step procedure to add the new panels we created to the wizard for the advanced store.
The following file contains the definitions for the panels to be displayed for the advanced store.

/usr/lpp/NetCommerce3/Tools/xml/nchs/store_creator/advanced.xml

Figure 105 on page 157, shows a sample from this file. Each <panel> tag is used to represent a separate screen in the wizard. It also defines its name, URL link of the screen, and helplink if required. You will see the two new panels (in bold) have been placed after the existing catalog panel. Insert these new entries and save the file.

```
 <panel name="storeCreatorPanelCatalog"
        url="/servlet/MerchantAdmin?DISPLAY=CTnchs.sc.advanced.Catalog"
        helplink="CTnchs.sc.advanced.Catalog.Help" />
<panel name="storeCreatorPanelCategory"
url="/servlet/DisplayPagesServlet?xmlfile=category.xml&amp;jspfile=/EXT_pages/html/Di
splayPages.jsp" />
<panel name="storeCreatorPanelProduct"
url="/servlet/DisplayPagesServlet?xmlfile=product.xml&amp;jspfile=/EXT_pages/html/Dis
playPages.jsp" />
<panel name="storeCreatorPanelBannerPage"
        url="/servlet/MerchantAdmin?DISPLAY=CTnchs.sc.basic.BannerPage"
        helplink="CTnchs.sc.basic.BannerPage.Help" />
```

Figure 105. Adding a new panel

Then we need to define the names that will appear on the left hand menu and the top banner bar when these screens are shown. These values are held in the properties file for the store creator. Open the following jar file using jar from a command line or any zip program.

/usr/lpp/NetCommerce3/Tools/lib/nchs.jar

And add new entries in the following properties file:

/com/ibm/commerce/tools/nchs/properties/StoreCreatorBasicNLS.properties

Or whatever is your language version. You will see a number of other language files. Figure 106 shows in bold the additions to make, where storeCreatorPanelProduct is the left hand menu value and storeCreatorPanelProduct_title is the banner bar title value.

```
storeCreatorPanelCatalog=Catalog
storeCreatorPanelCatalog_title=Product Catalog Information

storeCreatorPanelCategory_title=Category Pages Selection Screen
storeCreatorPanelCategory=Category Pages
storeCreatorPanelProduct_title=Product Pages Selection Screen
storeCreatorPanelProduct=Product Pages

storeCreatorPanelVerify=Verify
storeCreatorPanelVerify_title=Verify Your Choices
storeCreatorPanelModel=Store Model
storeCreatorPanelModel_title=Store Shopping Flow Model
```

Figure 106. StoreCreatorBasicNLS.properties

Then, re-create the jar file making a backup of the original. The simplest way from the command line is as follows:

```
jar cvf nchs.jar com/*
```

Stop the WebSphere Servlet Service by stopping the Web server.

```
stopsrc -s httpd
```

Copy the new jar file over the old jar remembering to make a backup.

```
cp nchs.jar /usr/lpp/NetCommerce3/lib/nchs.jar
```

Start the Websphere Servlet Service by restarting the Web server:

```
startsrc -s httpd
```

The new panels should appear as shown in Figure 107 below and Figure 108 on page 160.

Note

If the merchant changes their store style through the merchant tool to ExpressStoreModel and back again to ClassicStoreModel the store will be recreated with the default templates. You may either want to disable this option or add the category and product page choices to the merchant tool.

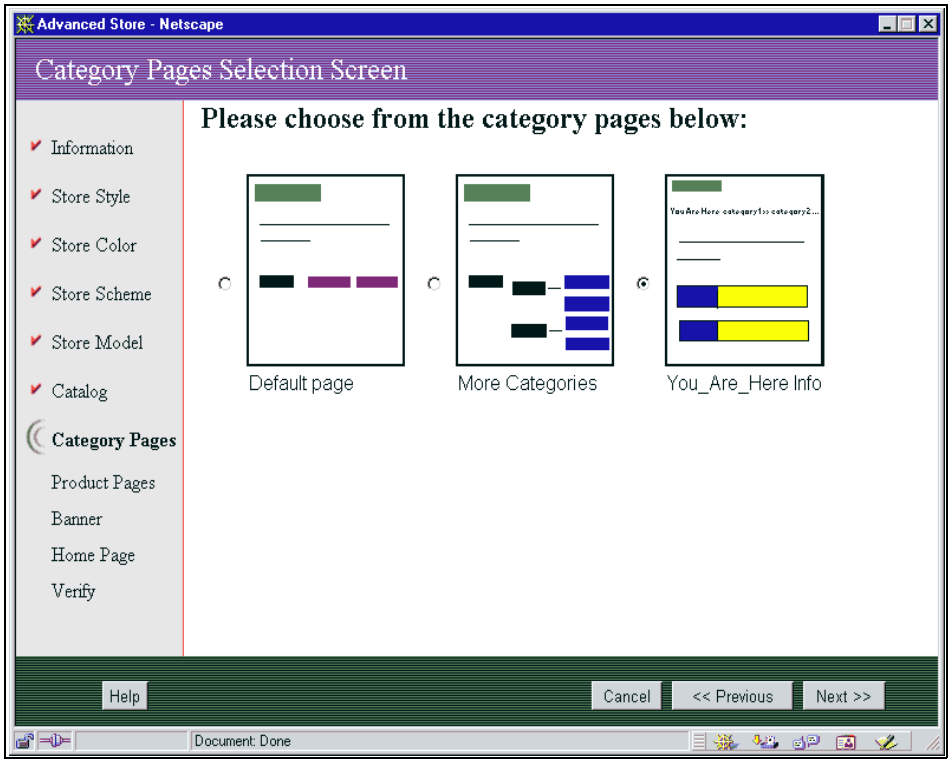


Figure 107. Store Wizard category pages panel

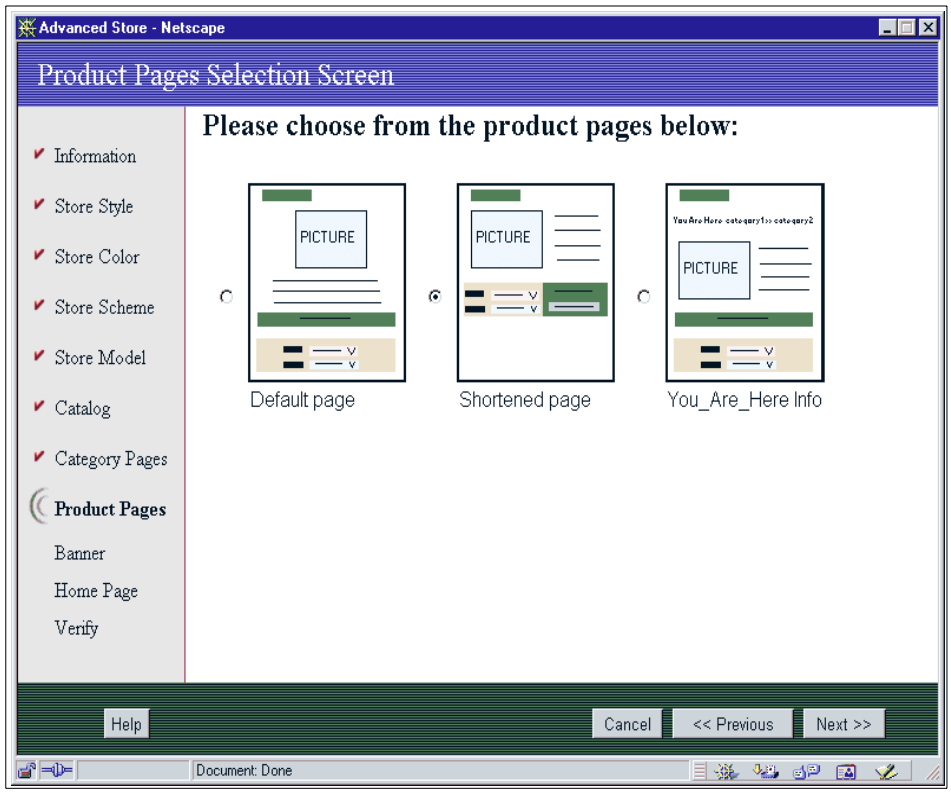


Figure 108. Store Wizard product pages panel

Chapter 9. Extended customization features

Even with the high number of features in WCS SPE, the ASP may need some customizations either because customers have special needs or because the ASP wants to provide special services to differentiate himself or herself from other ASPs. In this chapter, we will explore a number of customization examples and look at how it will impact the mall.

9.1 Shipping by product weight

The default configuration of WCS SPE does not include facilities to calculate shipping cost by weight although the standard facilities of the underlying WCS allow a rather flexible shipping cost system. The goal with this section is to give an example of how WCS SPE can be customized to handle shipping by weight in addition to the standard shipping by size. Before we actually start customizing the shipping system, let us take a brief look at the standard features.

In Figure 109, we have the screen for adding a new shipping method. In this example, we will add a method called IBM shipping.

Figure 109. Adding a new shipping method

For each shipping method, we can enter geographies or jurisdictions and then different shipping rates depending on destination. In the example in Figure 110 on page 162, we enter information for four locations, but all possible locations could be entered.

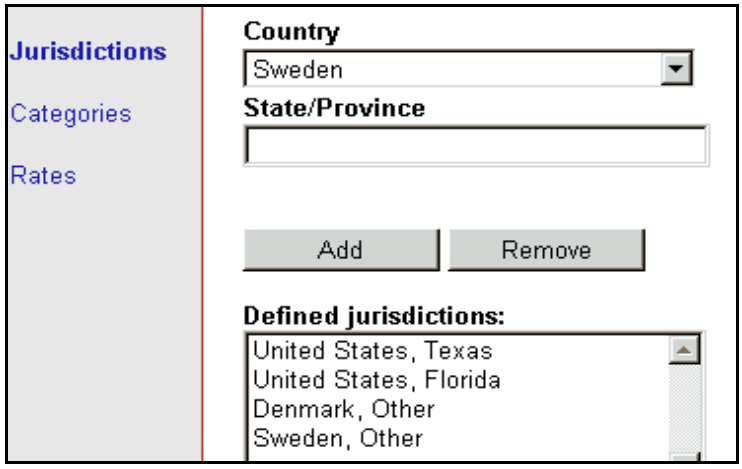


Figure 110. Add locations for which there are different shipping rates

After entering the locations, we define the shipping or item categories. At a later stage, we can assign a category to each item in the product catalog to use for the shipping cost calculation. The categories are shown in Figure 111.

Figure 111. Define shipping categories

Now, when the shipping method, destination, and item category are defined, we can enter the price grid for the shipping. Figure 112 on page 163 shows a sample table for the IBM shipping example.

	Cost/item	Large Items	Medium Items	Small Items	Weight_Based	Tiny_Items
Other Countries	7.0000	0.0000	0.0000	0.0000	1.0000	4.0000
United States, Alabama	1.0000	0.0000	0.0000	0.0000	1.0000	4.0000
United States, Texas	2.0000	0.0000	0.0000	0.0000	1.0000	4.0000
Denmark, Other	3.0000	0.0000	0.0000	0.0000	1.0000	4.0000
Sweden, Other	4.0000	0.0000	0.0000	0.0000	1.0000	4.0000
India, Other	5.0000	0.0000	0.0000	0.0000	1.0000	4.0000
Mexico, Other	6.0000	0.0000	0.0000	0.0000	1.0000	4.0000

Figure 112. Price grid for the shipping method

Figure 113 on page 164 shows a sample calculation for a Medium-sized item to Denmark. The shipping cost to Denmark for a Medium size item is zero USD plus three USD for shipping, which makes a total price of three USD for shipping. Figure 113 on page 164 shows a sample shipping cost calculation for a Tiny-sized item to Alabama USA.

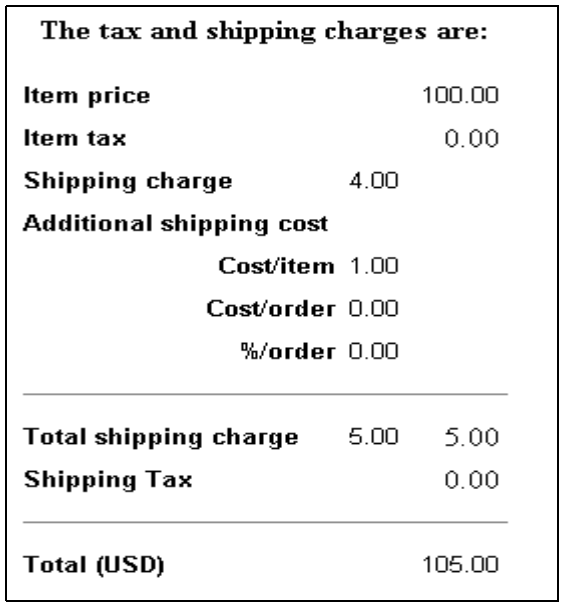


Figure 113. Sample calculation

Finally, we must define the shipping category for each product. We define one item as Medium, one item as Small and one as Tiny as shown in Figure 114 on page 165.

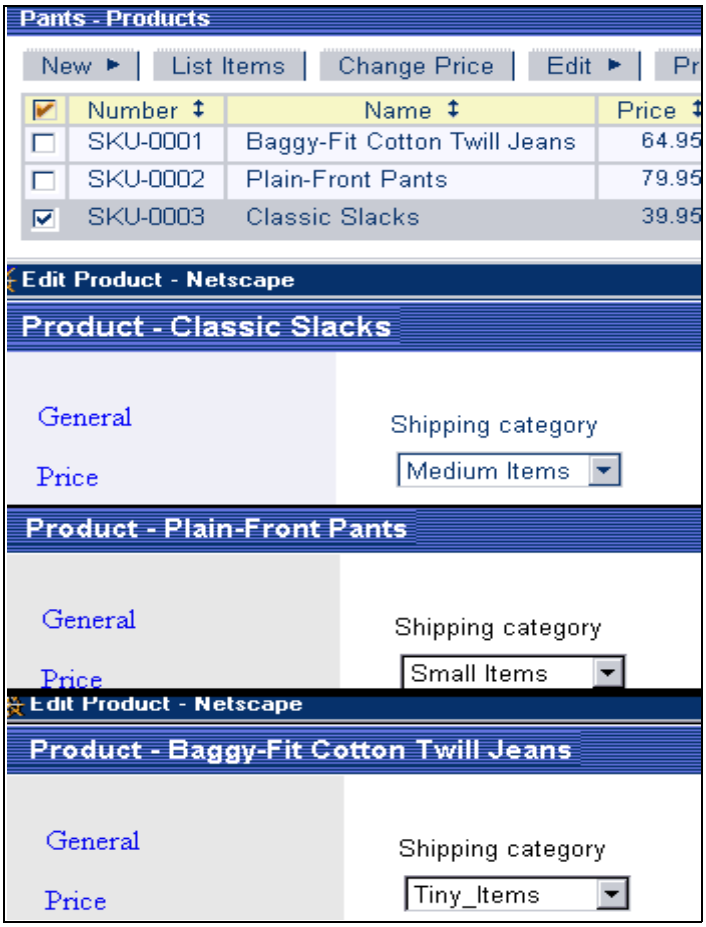


Figure 114. Define the shipping code with the merchant tool

Now, it's time to make a test buy. We buy three items: A medium, a small, and a tiny item, and the shipping address is in Austin TX, U.S.A. Select the IBM AIR shipping to see the shipping cost. We buy three items and pay three times the shipping cost per item plus shipping cost for a medium, small, and tiny item. The prices and shipping costs are $2.00 (times three), $0, $0, and 4$ and come to 10$. The prices are listed in Figure 112 on page 163.

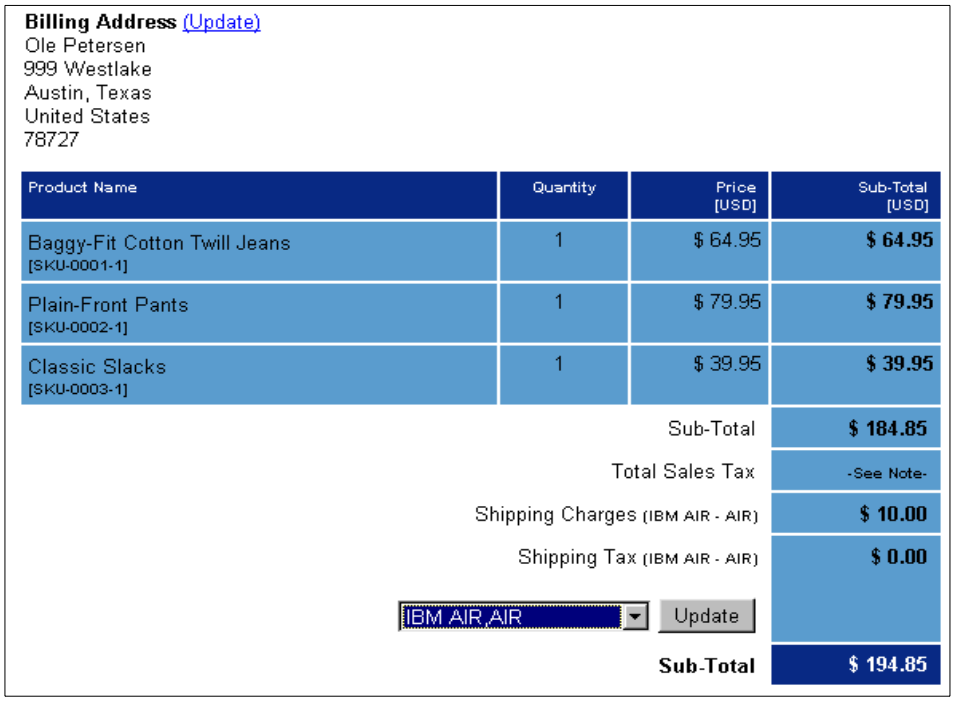


Figure 115. Presenting the purchase price for the customer

9.1.1 Shipping calculation in WCS SPE

Unlike WCS V3.2 and V4, WCS SPE is using a jurisdiction-based shipping solution. In other words, shipping rates are defined based on the following three factors: Shipping jurisdictions, shipping providers, and shipping categories (code). This is implemented by replacing the default Overridable Function (OF), GetOrdShTot_1.0, with another that does not support the shipping by weight calculation method. The shipping calculation is made using the following tables to set up the shipping rate for a specific merchant:

- SHIPJURST - Store the shipping country/state jurisdictions for each merchant (SPE used only).
- PRSPCODE - Store shipping code (WCS table).
- MSHIPMODE/SHIPMODE - Store the shipping providers for each merchant (WCS table).
- PSHIPRULE - Define the shipping rate for each combination of shipping jurisdiction, shipping code, and shipping providers (SPE used only).

All these tables are filled in by the SPE shipping wizard, and merchants are allowed to add/delete the shipping providers (see Figure 109 on page 161), add/delete shipping jurisdictions (see Figure 110 on page 162), and add/delete shipping categories (see Figure 111 on page 162). The shipping rates page will create a three-dimensional view of shipping rates based on the above three factors. One example for a shipping provider is shown in Figure 112 on page 163. A merchant will be allowed to modify the shipping rates from GUI and store the changes back to the PSHIPRULE table and other shipping-related tables.

Again, the shipping calculation for the SPE version is different from the base WCS product. It was designed to be the best fit for a net commerce hosting scenario (Ease of use from the GUI, merchant-specific shipping rate, shipping jurisdiction support, easy setup, and so on). These are the reasons why the SPE does not use the SHIPPING table and function from the base Net.Commerce version.

The SPE shipping wizard supports shipping categories, such as large, small, and medium items, and so on. However, in order to simplify the administration, the shipping wizard does not support shipping charges based on weight ranges since this would be too complicated to show in a table. For example, for each shipping category, you may have four or more different weight ranges: Zero to one kilograms, One to two kilograms, two to five kilograms and more than five kilograms. This means that you would have to show the following four-dimensional view from the GUI: Shipping providers, shipping categories, shipping jurisdictions, and shipping ranges. This would make the GUI hard to use, especially for those hosting merchants.

9.1.2 Database tables

Now, let us take a look at the underlying DB2 tables and see what information we have. The Data Model for shipping looks like that shown in Figure 116 on page 168.

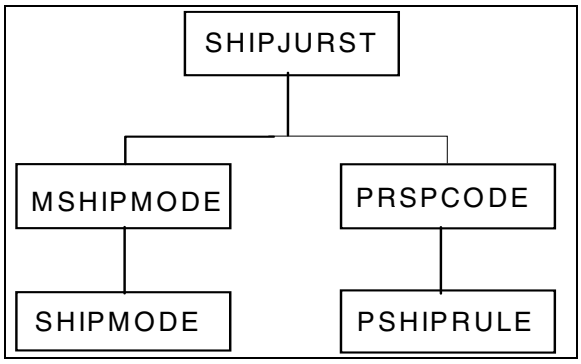


Figure 116. Shipping data model

Table 6 contains information about shipping jurisdictions. The maximum number of rows in this table is 100.

Table 6. SHIPJURST Table Shipping Jurisdiction

Column name	Column type	Column description
SPJUNBR	INTEGER NOT NULL	Shipping jurisdiction reference number. Unique reference number for this table.
SPJCNTRY	CHAR (25)	Country of shipping jurisdiction.
SPJSTATE	CHAR (25)	State of shipping jurisdiction.
SPJMENBR	INTEGER	Merchant reference number. This is a foreign key that references column MERFNBR in table MERCHANT.

Figure 117 on page 168 shows an example of the shipjurst table.

SPJUNBR	SPJCNTRY	SPJSTATE	SPJMENBR
677	United States	Alabama	854
678	United States	Texas	854
679	Denmark		854
680	Sweden		854
681	India		854
682	Mexico		854

Figure 117. SHIPJURST table

Table 7 contains the shipping codes that a merchant can assign to a product. The shipping code can be used to determine the method for calculating the shipping charges. The Net.Commerce system provides an Overridable Function that supports eight shipping charge calculation options. These options are described in the PSSPMTHD column. You can change the interpretation of the codes by programming an Overridable Function. Each row contains a shipping charge calculation method for a merchant.

Table 7. PRSPCODE: Product Shipping Code

Column name	Column type	Column description
PSRFNBR	INTEGER NOT NULL	Product shipping code reference number. This is a primary key.
PSCODE	CHAR (5) NOT NULL	Product shipping code. Together with PSMENBR and PSSPMTHD, this is a unique index.
PSMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references column MERFNBR in table MERCHANT. Together with PSCODE and PSSPMTHD, this is a unique index.
PSSPMTHD	CHAR (2)	Shipping charge calculation method. Together with PSMENBR and PSSPMTHD, this is a unique index. The default Overridable Function recognizes the methods Q1-Q4 and W1-W4 described below. for a further description of Q1-Q4 and W1-W4 see Table 17
PSSPDESC	VARCHAR (254)	Shipping code description.
PSFIELD1	VARCHAR (254)	Reserved for merchant customization.

Figure 118 shows an example of the PRSPCODE table.

```
PSRFNBR | PSCODE | PSMENBR | PSSPMTHD| PSSPDESC    | PSFIELD1 |
-------------------------------------------------------------------
 7      | G      | 2067    | Q3      |             |          |
-------------------------------------------------------------------
 323    | 1      | 7285    | Q3      | Large Items |          |
-------------------------------------------------------------------
 324    | 324    | 7285    | Q3      | Medium Items|          |
-------------------------------------------------------------------
 325    | 325    | 7285    | Q3      | Small Items |          |
-------------------------------------------------------------------
 326    | 326    | 7285    | Q3      | Tiny_Items  |          |
-------------------------------------------------------------------
```

Figure 118. PRSPCODE table listing

Table 8 indicates the shipping modes that each merchant supports. Each row associates a shipping mode with a merchant.

Table 8. MSHIPMODE

Column name	Column type	Column description
MMRFNBR	INTEGER NOT NULL	Merchant shipping mode reference number. This is a primary key.
MMSMNBR	INTEGER NOT NULL	Shipping mode reference number. This is a foreign key that references column SMRFNBR in table SHIPMODE.
MMMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references column MERFNBR in table MERCHANT.
MMDEFFS	TIMESTAMP (for DB2)	The date that the shipping mode becomes available
MMDEFFF	TIMESTAMP (for DB2)	The date that the shipping mode becomes unavailable.
MMFIELD1	VARCHAR (254)	Reserved for merchant customization.
MMFIELD2	INTEGER	Reserved for merchant customization.
MMTRKSPNBR	VARCHAR (64)	Carrier-assigned shipper number. Reserved for future use.

Figure 119 on page 170 shows an example of the MSHIPMODE table.

```
MMRFNBR| MMSMNBR| MMMENBR| MMDEFFS| MMDEFFF| MMFIELD1| MMFIELD2| MMTRKSPNBR
-------------------------------------------------------------------------------
323    | 319    | 7285   |        |        |         |         |
-------------------------------------------------------------------------------
2413   | 2409   | 7287   |        |        |         |         |
-------------------------------------------------------------------------------
2414   | 2410   | 7287   |        |        |         |         |
-------------------------------------------------------------------------------
```

Figure 119. mshipmode table example

Table 9 contains the names of the shipping carriers and the shipping service arrangements (shipping modes) that each carrier provides. Each row associates one shipping mode with one carrier.

Table 9. SHIPMODE

Column name	Column type	Column description
SMRFNBR	INTEGER NOT NULL	Shipping mode reference number. This is a primary key.

Column name	Column type	Column description
SMCARRID	CHAR (30)	Carrier identifier, such as Federal Express.
SMSPMODE	CHAR (30)	Carrier service shipping mode, such as FedEx Express Overnight.
SMSPDESC	VARCHAR (254)	Shipping mode description.
SMFIELD1	VARCHAR (254)	Reserved for merchant customization.
SMFIELD2	INTEGER	Reserved for merchant customization.
SMTRKNAME	VARCHAR (64)	Tracking application name. Reserved for future use.
SMTRKURL	VARCHAR (64)	Tracking application URL. Reserved for future use.
SMTRKSH	VARCHAR (64)	Tracking application socks host, if needed. Reserved for future use.
SMTRKSP	INTEGER	Tracking application socks port, if needed. Reserved for future use.
SMTRKICON	VARCHAR (64)	Tracking application icon for status link. Reserved for future use.
SMTRKTYPE	CHAR (8)	Tracking application inquiry type. Reserved for future use.

Figure 120 on page 172 show an example of the SHIPMODE table.

SMRFNBR	SMCARRID	SMSPMODE	SMSPDESC	SMFIELD1	SMFIELD2
1	Metropolitan	Overnight	Courier		
2	Metropolitan	2-Day	Courier		
3	Metropolitan	Ground	Courier		
319	IBM AIR	AIR	Courier		
2409	UPS	gift shipping	Courier		
2410	UPS	normal	Courier		

SMTRKNAME	SMTRKURL	SMTRKSH	SMTRKSP	SMTRKICON	SMTRKTYPE

Figure 120. shipmode table

The PSHIPRULE table in Figure 121 lists the shipping rates for each jurisdiction and shipping method. Compare it to the data entered in Figure 112.

SPRL NBR	SPME NBR	SPR TYPE	SP RULRG	SPCNTRY	SPSTATE	SP CITY	SP ZIPC	SPMT NBR	SPSRVS	SPFLTRT	SPTB NBR
323	7285	0	0	U.S.A.	Texas				319	1.00000	
324	7285	1	0	U.S.A.	Texas				319	0.00000	
325	7285	2	0	U.S.A.	Texas				319	0.00000	
326	7285	3	0	U.S.A.	Texas			323	319	1.00000	
327	7285	3	0	U.S.A.	Texas			324	319	2.00000	
328	7285	3	0	U.S.A.	Texas			325	319	3.00000	
329	7285	3	0	U.S.A.	Texas			326	319	4.00000	

Figure 121. PSHIPRULE table, Publish Shipping Rate Rule Table

9.1.3 Planning shipping by weight

In the previous section, we looked at the shipping system configured with WCS SPE and saw that it cannot support shipping by product weight, and we must, therefore, plan an extension.
The shipping cost is calculated by the process task, GET_ORD_SH_TOT, which, in turn, by default, calls the default Overridable Function, GetOrdShTot. This function gets the order reference number, ORDER_REF_NUM, as a string and delivers product shipping cost as a string. The task is to replace the default function with a new one, which can calculate the cost by weight. As described earlier in this chapter, there are many good reasons why the standard method does not support shipping by weight, and we must make some assumptions to simplify system maintenance before we can design an extension to handle shipping by weight.

The target for the WCS SPE product is small- and medium-size merchants who want to go on the Internet with a professional Web shop that can handle payment and shipping. The assumption is that these kinds of merchants most likely want to ship by units or by weight, not a combination of the two. In other words, when a given merchant goes on the Web, all his or her invoices will be by units or by weight. This simplifies the calculations a lot if we do not have to make mixed invoices because we can assume that the order will be quantum-based if just one item in the order is shipped by quantum. Also, more importantly, it makes it much easier for the merchant because he or she has only to maintain one type of price table: Units or weight. We will also assume that this type of merchant, if shipping by weight, will use one and only one shipping provider; this will make it simpler for the merchant to maintain the shipping cost tables, and, again, in this market segment, it is important to make it easy to maintain even if we have to sacrifice some flexibility or options. With these restrictions, the merchant can type his or her shipping cost into a table, very much like the one used for shipping by units. The table might look like Table 10.

Table 10. Sample shipping by weight cost

	0-0.5 weight unit	**0.5-1 weight unit**	**1-2 weight unit**	**2-- weight unit**
United States Alabama				
United States Alaska				
United States Arizona				

	0-0.5 weight unit	0.5-1 weight unit	1-2 weight unit	2-- weight unit
More states ..				
Afghanistan				
Albania				
Algeria				
Andorra				
More countries ...				

To calculate the shipping cost, the calculation function gets the order reference number as an input parameter, and, from this number, it has to resolve all order lines and the destination. The Database table look up could appear as shown in Figure 122 on page 174.

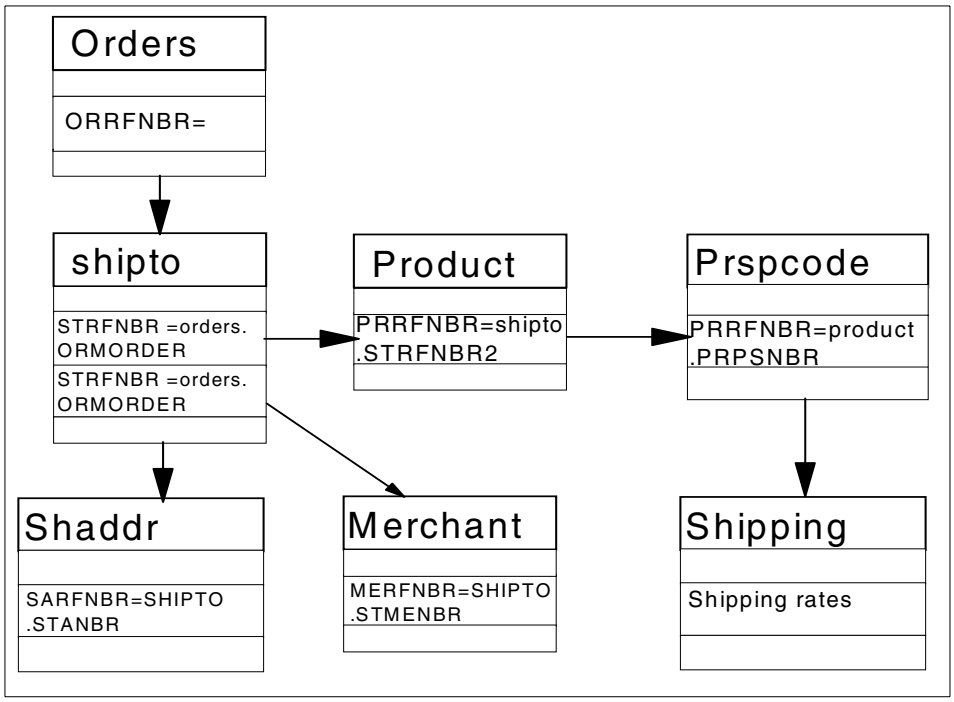


Figure 122. Order processing

Let us take a look at how to find the shipping cost from the database. The ORDERS table shown in Table 11 contains information about orders that are placed by shoppers. Each row corresponds to a single order, and each order has a line in the SHIPTO table for each item in the order. The lines in the SHIPTO table point to the PRODUCT table where we can get information about the product such as product name, product number, and weight. From the PRODUCT table, we can link to the PRSPCODE table where we get information about the shipping mode. The default shipping mode in WCS SPE is Q3, but, in this example, we will work with shipping mode W3 for weight-based shipping in addition to the Q3 mode. If the shipping is in W3 mode, we can link from the PRSPCODE table to the SHIPPING table and get the shipping cost. Finally, for each line in the shipto table, we have a pointer to the shopper address and merchant information.

Table 11. Orders

Column name	Column type	Column description
ORRFNBR	INTEGER NOT NULL	Unique internally generated order reference number. This is a primary key.
ORSHNBR	INTEGER NOT NULL	Shopper reference number. This is a foreign key (delete rule = no action) that references SHRFNBR in table SHOPPER.
ORMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references column MERFNBR in table MERCHANT.
ORMORDER	CHAR (30)	Order reference number generated by the merchant. Unique within a merchant store.
ORBLLTO	INTEGER	Address reference number for the billing address. This is a foreign key (delete rule = no action) that references column SARFNBR in table SHADDR.
ORPRTOT	NUM(15,2)	Total product price for this order.
ORTXTOT	NUM(15,2)	Total sales tax for this order.
ORSHTOT	NUM(15,2)	Total shipping charges for this order.
ORSHTXTOT	NUM(15,2)	Total tax on shipping charges for this order.
ORCPCUR	CHAR (10)	Currency in which the price is expressed. The format of the currency must adhere to ISO 4217 standards.

Column name	Column type	Column description
ORSTAT	CHAR (1)	Order status: P - Order in pending state C - Order in completed state (order was placed) X - Order was cancelled I - Inventory updated pending (order no longer pending) M - Ready for authorization (order passed inventory update) All single upper case letters are reserved for Net.Commerce.
ORLOCK	CHAR(1)	Lock indicator for disallowing any updates to the order.
ORPSTMP	TIMESTAMP	(for DB2) The date and time the order was placed.
ORUSTMP	TIMESTAMP (for DB2)	The date and time the order entry was last updated.
ORFIELD1	INTEGER	Reserved for merchant customization.
ORFIELD2	NUM (15,2)	Reserved for merchant customization.
ORFIELD3	VARCHAR (254)	Reserved for merchant customization.

An example of the content in the ORDERS table is shown in Figure 123.

```
 ORRFNBR | ORSHNBR | ORMENBR | ORMORDER | ORBLLTO | ORPRTOT | ORTXTOT |
-------------------------------------------------------------------------
 2206    | 0       | 7285    | 2206     | 46 | 64.95   | 0.00    |
-------------------------------------------------------------------------


 ORSHTOT | ORSHTXTOT | ORCPCUR | ORSTAT | ORLOCK | ORPSTMP| ORUSTMP
-------------------------------------------------------------------------------
 14.80   | 0.00      | USD     | P      | 1      |        | 2000-03-28
         |           |         |        |        |        | 14:38:18.538445
-------------------------------------------------------------------------------


ORFIELD1 | ORFIELD2 | ORFIELD3
-----------------------------
         |          |
-----------------------------
```

Figure 123. Content of ORDERS table

The SHADDR table shown in Table 12 on page 177 serves as an address book for each registered shopper. At the time of registration, the shopper

provides his or her address, and this entry is flagged as permanent. When a shopper moves, the shopper can provide a new address, and a new entry is added to the table. The old address is not discarded, but it is flagged as temporary. Temporary rows are also created if a shopper provides a new address for a specific order without updating the address book. Shoppers can also add addresses for other individuals, such as relatives or places to this table. All such entries are flagged as permanent.

Table 12. Selected columns from the SHADDR table

Column name	Column type	Column description
SARFNBR	INTEGER NOT NULL	Unique address reference number - internally generated. This is a primary key.
SASHNBR	INTEGER NOT NULL	Shopper reference number. This is a foreign key that references the SHRFNBR column in the SHOPPER table.
SANICK	CHAR (31) NOT NULL	This is the nickname of another individual, such as a relative, to whom the address information applies.
SATITLE	CHAR (5)	Individual's title: Dr. Mr. Mrs. Ms. N - Not provided (default)
SALNAME	CHAR (30)	Individual's last name.
SAFNAME	CHAR (30)	Individual's first name.
SAPHONE1	CHAR (30)	Individual's phone number 1 (For example, daytime phone).
SAADDR1	CHAR (50) NULL	Individual's address line 1.
SAADDR2	CHAR (50)	Individual's address line 2.
SAADDR3	CHAR (50)	Individual's address line 3.
SACITY	CHAR (30) NULL	Individual's city name.

Column name	Column type	Column description
SASTATE	CHAR (20) NULL	Individual's state, province, or equivalent, abbreviated.
SACNTRY	CHAR (30) NULL	Individual's country.
SAZIPC	CHAR (20)	Individual's zip code or equivalent.
SASTMP	TIMESTAMP (for DB2)	The date and time the row was created.

Figure 124 shows one data record in the shopper address, SHADDR table.

```
SARFNBR | SASHNBR | SANICK       | SATITLE | SALNAME| SAFNAME| SAPHONE1
-------------------------------------------------------------------------
 46      | 316     | hans@ibm.com |         | hansen | Jan    | 977-0983
-------------------------------------------------------------------------

SAADDR1      | SAADDR2| SAADDR3| SACITY| SASTATE| SACNTRY | SAZIPC| SASTMP
-------------------------------------------------------------------------------
999 Parmer Ln|        |        | Austin| Texas  | USA     | 78777 |
-------------------------------------------------------------------------------
```

Figure 124. Data in selected columns in SHADDR table

The SHIPTO table shown in Table 13 associates each product and item in a suborder with a shipping address. Each row corresponds to one product or item.

Table 13. Ship to table description

Column name	Column type	Column description
STRFNBR	INTEGER NOT NULL	Unique shipto reference number, internally generated. This is a primary key.
STORNBR	INTEGER	Order reference number. This is a foreign key that references the ORRFNBR column in the ORDERS table.
STSANBR	INTEGER NOT NULL	Address reference number for the shipping address.
STSHNBR	INTEGER NOT NULL	Shopper reference number. This is a foreign key that references the SHRFNBR column in the SHOPPER table.
STMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references the MERFNBR column in the MERCHANT table.

Column name	Column type	Column description
STPRNBR	INTEGER NOT NULL	Product reference number. This is not a foreign key. Note that no foreign key is defined in the STPRNBR column so that no information on any order is lost when the product in the PRODUCT table is deleted.
STPRICE	NUM(15,2)	Unit price of the item.
STCPCUR	CHAR (10)	Currency in which the price is expressed. The format of the currency must adhere to ISO4217 standards.
STPCODE	CHAR (5)	The date and time the shipto entry was made.
STCMT	VARCHAR (254)	Comments from shopper, such as a greeting for a gift.
STQUANT	INTEGER NOT NULL	Quantity ordered.
STSTAT	CHAR (1) NOT NULL	Order Status: P - In pending state C - In past state X - Cancelled I - Inventory update pending (shipto no longer pending) M - Ready for authentication (shipto passed inventory update)
STPSTMP	TIMESTAMP (for DB2)	The date and time the shipto entry was made.
STUSTMP	TIMESTAMP (for DB2)	The date and time the shipto entry was last updated.
STSMNBR	INTEGER	Merchant shipping mode reference number (refer to MSHIPMODE).
STFIELD1	INTEGER	Reserved for merchant customization.
STFIELD2	VARCHAR (254)	Reserved for merchant customization.
STTRKNBR	CHAR (64)	Tracking number. Reserved for future use.
STTRKDATE	DATE	Shipment date. Reserved for future use.
STBASEPRICE	NUM (15, 2)	If the product price was converted, this is the original price before conversion.
STBASECURR	CHAR (3)	Original currency before conversion in alphabetic code as per ISO 4217.

Figure 125 shows an example of data in the shipto table. The STORNBR field references ORRFNBR in the ORDERS table. STSANBR is the Address reference number for the shipping address. STSHNBR references SHRFNBR in the SHOPPER table; STMENBR references MERFNBR in the MERCHANT table, and STPRNBR references product number in the PRODUCT table.

```
STRFNBR | STORNBR | STSANBR | STSHNBR | STMENBR | STPRNBR | STPRICE | STCPCUR
-------------------------------------------------------------------------------
1896    | 1892    | 2538    | 2538    | 7285    | 492     | 39.95   | USD
-------------------------------------------------------------------------------
1899    | 1895    | 2495    | 316     | 7285    | 485     | 64.95   | USD
-------------------------------------------------------------------------------
1944    | 1895    | 2495    | 316     | 7285    | 489     | 79.95   | USD
-------------------------------------------------------------------------------


STPCODE | STCMT     | STQUANT | STSTAT | STPSTMP         | STUSTMP
-----------------------------------------------------------------
default | Comments: | 1       | C      | 2000-03-23      | 2000-03-23
        |           |         |        | 17:54:32.252073 | 17:59:08.829404
-----------------------------------------------------------------
default | Comments: | 1       | P      | 2000-03-24      | 2000-03-27
        |           |         |        | 17:26:07.698597 | 11:04:29.584062
-----------------------------------------------------------------
default | Comments: | 1       | P      | 2000-03-24      | 2000-03-27
        |         : |         |        | 17:26:25.590922 | 11:04:29.590862
-----------------------------------------------------------------

STSMNBR| STFIELD1| STFIELD2| STBASEPRICE| STBASECURR| STTRKNBR| STTRKDATE
-------------------------------------------------------------------------
323    |         |         |            |           |         |
-------------------------------------------------------------------------
323    |         |         |            |           |         |
-------------------------------------------------------------------------
323    |         |         |            |           |         |
-------------------------------------------------------------------------
```

Figure 125. Shipto table content

The MERCHANT table (selected columns of which are shown in Table 14) describes each merchant as well as information on the primary contact for the merchant. Each row corresponds to one merchant. For a one-merchant mall, this table contains only one row. No foreign key is enforced for this table.

Table 14. Selected Columns from the MERCHANT table

Column name	Column type	Column description
MERFNBR	INTEGER NOT NULL	Unique merchant reference number - internally assigned. This is a primary key.

Column name	Column type	Column description
MENAME	CHAR (80)	Merchant's company name. If no name is provided when the store is created, this column contains the store name of the merchant as defined in the MESTNAME column in this table.
MECLNAM	CHAR (30) NOT NULL	Merchant contact's last name.
MECFNAM	CHAR (30) NOT NULL	Merchant contact's first name.
MECPH1	CHAR (30) NOT NULL	Merchant contact's primary phone number.
MEPHONE	CHAR (30) NOT NULL	Merchant's company phone number.
MEADDR1	CHAR (50)	Merchant's company street address line 1.
MECITY	CHAR (30) NOT NULL	Merchant's company city name.
MESTATE	CHAR (20) NOT NULL	Merchant's company state, province, or equivalent (abbreviated).
MECNTRY	CHAR (30) NOT NULL	Merchant's company country.
MECUR	CHAR (10) NOT NULL	Currency in which prices for this store are expressed. The format of the currency must adhere to ISO 4217 standards.

Figure 126 on page 182 show data from two merchants in the merchants table.

```
MERFNBR| MENAME          | MECLNAM | MECFNAM| MECPH1
-------------------------------------------------------------
 1001   | Concept Store 1| Fong    | James  | (555)123-1001
-------------------------------------------------------------
 7285   | shop4          | Hansen  | Per    | 512-977-0999
-------------------------------------------------------------

| MEPHONE        | MEADDR1
-----------------------------------
| 1-888-4KJ-HOLD | 895 Don Mills Rd.
-----------------------------------
|                | 3401 Parmer LN
-----------------------------------

| MECITY     | MESTATE | MECNTRY       | MECUR
-----------------------------------------------
| North York | ON      | Canada        | CAD
-----------------------------------------------
| Austin     | tx      | United States | USD
-----------------------------------------------
```

Figure 126. Selected rows from the Merchant table

The PRODUCT table describes all the items available at all stores. An item is a product that must be qualified by one or more attributes to be resolved into a Stock Keeping Unit (SKU). A SKU is an orderable item. In this table, items must have a non-null value in the PRPRFNBR column. Each row contains information on one product item.

Table 15. PRODUCT table

Column name	Column type	Column description
PRRFNBR	INTEGER NOT NULL	Product reference number. This is a primary key.
PRMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references the MERFNBR column in the MERCHANT table. Together with the PRNBR column, this is a unique index
PRPRFNBR	INTEGER	Product number of the parent. This is a foreign key that references the PRRFNBR column in the PRODUCT table. If null, this is the topmost product.
PRNBR	CHAR (64) NOT NULL	Product number or item SKU. Together with the PRMENBR column, this is a unique index.
PRSDESC	VARCHAR (254)	Short description of the product including its name.

Column name	Column type	Column description
PRLDESC1	LONG VARCHAR (for DB2)	Detailed description 1 of the product.
PRLDESC2	LONG VARCHAR (for DB2)	Detailed description 2 of the product.
PRLDESC3	LONG VARCHAR (for DB2)	Detailed description 3 of the product.
PRTHMB	CHAR (254)	Product thumbnail image file path and name.
PRFULL	CHAR (254)	Product full-sized image file path and name.
PRPUB	SMALLINT	Should this product be displayed to the public? 0 -No 1 - Yes 2 - Marked for deletion
PRKNUTAG	CHAR (64)	Reserved for IBM use.
PRWGHT	NUM (15,4)	Weight of the item for determining the shipping charges. The default is 0.
PRWMEAS	CHAR (20)	Unit of measurement for the weight (for example, kilograms or ounces).
PRLNGTH	NUM (15,4)	Length of the item (can be used for shipping). The default is 0.
PRWIDTH	NUM (15,4)	Width of the item (can be used for shipping). The default is 0.
PRHEIGHT	NUM (15,4)	Height of the item (can be used for shipping). The default is 0.
PRSMEAS	CHAR (20)	Unit of measurement for the dimensions (for example, meters or inches).
PRPSNBR	INTEGER Product	Shipping code for shipping charges. The default is null. This is not a foreign key.
PRPCODE	CHAR (25)	No field description available
PRURL	VARCHAR (254)	URL for soft goods or links.
PRVENT	INTEGER	Number of items in stock. The default is null, meaning the merchant has not yet stocked the product. This field is not used for a product that has items associated with it.

Column name	Column type	Column description
PRAVDATE	TIMESTAMP (for DB2)	Availability date of the product. If the item is out of stock or is not yet stocked, the merchant can specify the availability date.
PRSPECIAL	CHAR (4)	Special information about the product: S - on sale
PRSTMP	TIMESTAMP (for DB2)	The date and time the product information was last updated.
PRFIELD1	INTEGER	Reserved for merchant customization.
PRFIELD2	INTEGER	Reserved for merchant customization.
PRFIELD5	VARCHAR (254)	Reserved for merchant customization.
PRFIELD4	VARCHAR (254)	Reserved for merchant customization.
PRFIELD3	NUM (15,2)	Reserved for merchant customization.
PRFIELD4	VARCHAR (254)	Reserved for merchant customization.
PRFIELD5	VARCHAR (254)	Reserved for merchant customization.
PRDCONBR	INTEGER	Product discount code. This is a foreign key that references the DCORFNBR column in the DISCCODE table.
PROID	CHAR (36)	Reserved for IBM use.

Figure 127 on page 185 and Figure 128 on page 186 show data examples from the PRODUCT table.

PRRFNBR	PRMENBR	PRPRFNBR	PRNBR	PRSDESC
109	6001		9545F0G	ThinkPad 755C
486	7285	484	SKU-0001-2	Baggy-Fit Cotton
489	7285	487	SKU-0002-2	Plain-Front Pants
492	7285	490	SKU-0003-2	Classic Slacks
1018	7801		SKU-0008	Petunia Thunderstorm

PRLDESC1	PRLDESC2	PRLDESC3	PRTHMB	PRFULL	PRPUB
			/TP755CE.GIF	TP755CE.GIF	1
Comfor ...				pants2.gif	1
Polyest..				pants1.gif	1
Perfect...				shorts2.gif	1
A bolt ...				flower_8L.jpg	1

Figure 127. PRODUCT table (1 of 2)

```
PRKNUTAG | PRWGHT  | PRWMEAS | PRLNGTH | PRWIDTH | PRHEIGHT | PRSMEAS | PRPSNBR
---------------------------------------------------------------------------------
         | 0.0000  |         | 0.0000  | 0.0000  | 0.0000   |         |
---------------------------------------------------------------------------------
         |         |         |         |         |          |         | 326
---------------------------------------------------------------------------------
         |         |         |         |         |          |         | 325
---------------------------------------------------------------------------------
         |         |         |         |         |          |         | 324
---------------------------------------------------------------------------------
         |         |         |         |         |          |         |
---------------------------------------------------------------------------------

PRPCODE | PRURL | PRVENT | PRAVDATE             | PRSPECIAL | PRSTMP
-------------------------------------------------------------------------------
--
        |       |        |                      |           | 2000-03-20 10:03:22.0
-------------------------------------------------------------------------------
--
default |       | 45     | 2000-03-20 17:03:28.1|           | 2000-03-23 17:04:30.7
-------------------------------------------------------------------------------
--
default |       | 232    | 2000-03-20 17:03:28.4|           | 2000-03-23 17:02:00.9
-------------------------------------------------------------------------------
--
default |       | 76     | 2000-03-20 17:03:28.8|           | 2000-03-23 17:02:43.3
-------------------------------------------------------------------------------
--
default |       | 34     | 2000-03-21 16:42:50.7|           | 2000-03-21 16:42:50.7
-------------------------------------------------------------------------------
--

PRFIELD1 | PRFIELD2 | PRFIELD3 | PRFIELD4 | PRFIELD5 | PRDCONBR | PROID
-----------------------------------------------------------------------
         |          |          |          |          |          |
-----------------------------------------------------------------------
         |          |          |          |          |          |
-----------------------------------------------------------------------
         |          |          |          |          |          |
-----------------------------------------------------------------------
         |          |          |          |          |          |
-----------------------------------------------------------------------
         |          |          |          |          |          |
-----------------------------------------------------------------------
```

Figure 128. PRODUCT table (2 of 2)

With the data in the product table, we have the weight of each item in the shipment and can calculate the total weight. To calculate the cost, we must

have the cost per weight unit; the SHIPPING table holds this information. Let us now take a look at the data in the SHIPPING table shown in Table 16.

Table 16. Shipping table description

Column name	Column type	Column description
SPRFNBR	INTEGER NOT NULL	Shipping reference number. This is a primary key.
SPMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that references the MERFNBR column in the MERCHANT table.
SPMMNBR	INTEGER	Merchant shipping mode reference number. This is a foreign key that references the MMRFNBR column in the MSHIPMODE table.
SPPSNBR	INTEGER	Product shipping code reference number. This is a foreign key that references the PSRFNBR column in the PRSPCODE table.
SPADDRJR	CHAR (20)	Address jurisdiction indicator. This is the jurisdiction on which the price is based.
SPCNTRY	CHAR (30)	Country identifier. This is the country on which the price is based.
SPSTRAMT	NUM (15,2) NOT NULL	Starting amount of a range. If used for a weight, the weight must be in the same units as defined for the product in the PRODUCT table.
SPENDAMT	NUM (15,2) NOT NULL	Ending amount of a range. If used for a weight, the weight must be in the same units as defined for the product in the PRODUCT table.
SPCHRGE	NUM (15,2)	Shipping charge.
SPDEFFF	TIMESTAMP (for DB2)	The date that the shipping calculation method is no longer valid.
SPFIELD1	VARCHAR (254)	Reserved for merchant customization.
SPFIELD2	NUM (15,2)	Reserved for merchant customization.
SPCURR	CHAR (3)	The currency of the amounts in the SPCHRGE and SPRATE columns. If NULL, then the value of MECUR in the MERCHANT table is used.
SPRATE	NUM (8,2)	Shipping rate.

Column name	Column type	Column description
SPDEFFS	TIMESTAMP (for DB2)	The date that the shipping calculation method becomes effective.

Figure 129 shows an example of data in the SHIPPING table. The SPSTRAMT, SPENDAMT, and SPCHRGE columns hold the lower weight boundary, the upper weight boundary, and the shipping charge for shipping to to this jurisdiction. The SPMMNBR and SPPSNBR fields link to the MSHPMODE and PRSPCODE tables.

```
SPRFNBR | SPMENBR | SPMMNBR | SPPSNBR | SPADDRJR | SPCNTRY | SPSTRAMT |
-------------------------------------------------------------------------
226     | 471     | 226     | 226     |          |         | 1.00     |
-------------------------------------------------------------------------

SPENDAMT   | SPCHRGE | SPRATE | SPDEFFS             | SPDEFFF                    |
-------------------------------------------------------------------------------------
9999999.00 | 5.00    |        | 2000-03-07 18:43:02.665560 | 9999-12-31 12:00:00.000000
-------------------------------------------------------------------------------------

SPFIELD1 | SPFIELD2 | SPCURR |
------------------------------
|       |          |        |
------------------------------
```

Figure 129. SHIPPING table example

```
SPRFNBR | SPMENBR | SPMMNBR | SPPSNBR | SPADDRJR | SPCNTRY | SPSTRAMT
-----------------------------------------------------------------------
46      | 7285    | 7       | 7       |          | CAD     |  0.00
-----------------------------------------------------------------------
47      | 7285    | 7       | 7       |          | CAD     | 10.00
-----------------------------------------------------------------------
48      | 7285    | 7       | 7       |          | CAD     | 20.00
-----------------------------------------------------------------------
49      | 7285    | 7       | 7       |          | CAD     | 30.00
-----------------------------------------------------------------------

SPENDAMT  | SPCHRGE | SPRATE | SPDEFFS | SPDEFFF | SPFIELD1 | SPFIELD2 | SPCURR
-------------------------------------------------------------------------------
10.00     | 3.00    |        |         |         |          |          |
-------------------------------------------------------------------------------
20.00     | 6.00    |        |         |         |          |          |
-------------------------------------------------------------------------------
30.00     | 10.00   |        |         |         |          |          |
-------------------------------------------------------------------------------
999999.00 | 15.00   |        |         |         |          |          |
-------------------------------------------------------------------------------
```

Figure 130. SHIPPING table example

The standard WCS system operates with 4 quantity and 4 weight based calculation methods, in the SPE version this is limited to only one method, quantum based Q3 method for the reasons described earlier. In this example however we will reenable one weight based calculation method. For compatibility it will be named W3, that is, we will have a Q3 and a W3 method in the system.

Table 17. Shipping charge calculation methods in the WCS system

Shipping method	Method description	Detailed description
Q1	Quantity/Cumulative/ Total	Cost calculated for total quantity of items within each specified range. $1.00 for one to five books, $2.00 for six to 10 books. Seven books will cost $3.00
Q2	Quantity/Cumulative/ Unit	Cost for each item,calculated for each specified range. $1.00 per book for one to five books, and $2.00 per book for six to 10 books. Seven books will cost $9.00
Q3	Quantity/Range/ Total	Cost for total quantity of items, falling within a specified range. $1.00 for one to five books, and $2.00 for six to 10 books. Seven books will cost $2.00.
Q4	Quantity/Range/ Unit	Cost for each item, based on total quantity falling within a specified range. $1.00 for one to five books, and $2.00 for six to 10 books. Seven books will cost $14.00
W1	Weight/Cumulative/ Total	Cost for total weight of shipment calculated for each specified range. $1.00 for one to five kg, $2.00 for six to 10 kg. Seven kg will cost $3.00
W2	Weight/Cumulative/ Unit	Cost for each unit of weight calculated for each specified range. $1.00 per kg for one to five kg, and $2.00 per kg for six to 10 kg. Seven kg will cost $9.00
W3	Weight/Range/ Total	Cost for total weight of shipment, falling within a specified range. $1.00 for one to five kg, and $2.00 for six to 10 kg. Seven kg will cost $2.00.
W4	Weight/Range/ Unit	Cost for each unit of weight, based on total weight within a specified range. $1.00 for one to five kg, and $2.00 for six to 10 kg. Seven kg will cost $14.00

Now, after we know which tables are involved in the shipping calculation, the steps needed to enable the calculation can be determined; the steps that must be completed are:

1. Add new features to the merchant tool interface to support the maintenance of the following tables:
   - SHIPPING table maintaining the rates per. jurisdiction
   - PRODUCT table maintaining the weight and weight unit per product
   - PRSPCODE table maintaining the psspmthd field to reflect weight based calculation
2. Write a new function to replace the standard GetOrdShTot_1.0 function.

The merchant must be able to maintain the SHIPPING, PRODUCT, and PRSPCODE tables, and it should be enabled through the merchant tool. The best way to do this is to implement some Net.Data macros. The function to replace the GetOrdShTot_1.0 function must be written in C++. The following sections will describe how to add these functions.

9.1.4 Adding new features to the merchant tool

The goal with the merchant tool enhancement is to add the ability to maintain three database tables from the tool. As mentioned in the previous section, the tables we want to maintain are the PRSPCODE, PRODUCT, and SHIPPING tables. We will make one WCS macro to maintain each table. This will add three menu items to the merchant tool; we add a new section as shown in Figure 131.

Figure 131. Customized merchant tool with the shipping by weight additions

To add these three extra menu items to the menu of the merchant tool and add functions, we will have to create three additional WCS macro files. in this example, the macro files are shipbywenable.d2w, shipbywdata.d2w, and shipbyweight.d2w. The macros are in the following directory:

/usr/lpp/NetCommerce3/macro/en_US/ncadmin/storemgr/

The source code can be found in Appendix E, “Shipping by weight source code” on page 339.

To activate these macros for the merchant tool, perform the following steps:

1. Locate and open the XML file, navigation.xml, for the navigation bar in the advanced store. The file is placed in the following directory: usr/lpp/NetCommerce3/Tools/xml/nchs/mtool/advanced/.
2. In the file, locate the line that reads <node name="storeSetup". Locate where the store setup section ends, and add the following new link tag just before the next section:

```
<node name="ShipByWeight"
         image="/NCTools/images/clear.gif"
        openImage="/NCTools/images/clear.gif"
         users="siteAdmin storeAdmin" >

       <node name="ShipByWeightenable"

url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/shipbywenable.d2w/r
eport?merfnbr=$env.merchant_id$"
        wholepage="true"
       image="/NCTools/images/clear.gif"
      users="siteAdmin storeAdmin catalogAdmin" />

       <node name="CostShipByWeight"

url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/shipbyweight.d2w/re
port?merfnbr=$env.merchant_id$"
        wholepage="true"
       image="/NCTools/images/clear.gif"
      users="siteAdmin storeAdmin catalogAdmin" />

       <node name="ShipByWeightData"

url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/shipbywdata.d2w/rep
ort?merfnbr=$env.merchant_id$"
wholepage="true"
         image="/NCTools/images/clear.gif"
         users="siteAdmin storeAdmin catalogAdmin" />

          </node>
```

where

- **node name** denotes the variable used to identify the link.
- **url** denotes the action of the link.
- **images** defines the image on the navigation bar to the left of the link.
- **users** defines which users are allowed to see this link.

3. Save the XML file.
4. Locate and open the file navigationNLS.properties. It is found in /usr/lpp/NetCommerce3/Tools/lib/nctools.jar. To access the properties file, you must unjar the file by issuing the following from the command line:
   ```
   jar -xvf nctools.jar
   ```
5. This command will unpack the jar file into two sub folders, called com and manifest. The navigationNLS. properties file can be found in the directory, /usr/lpp/NetCommerce3/Tools/lib/com/ibm/commerce/tools/nchs/properties
6. Open the navigationNLS.properties file using a text editor and add the following lines:
   ShipByWeight = Shipping by weight
   ShipByWeightenable = Enable/disable shipping by weight
   CostShipByWeight = Ship cost by weight
   ShipByWeightData = Product weight data

   Save the file.
7. Re-create the nctools.jar file by issuing the following command to pack the file from the command line:
   ```
   jar -cf0 nctools.jar com
   ```
8. Restart WCS and Lotus Domino Go Web sever to make the changes reflected.

Now, after implementing these changes, we are able to maintain the PRODUCT, PRSPCODE, and SHIPPING tables. We can see the changes when we open the merchant interface as shown in Figure 132 on page 193.

Figure 132. Customized shipping by weight interface

Three new functions are added to the menu:

- Enable/disable shipping by weight
- Ship cost by weight
- Product weight data

The first function, Enable/disable shipping by weight, updates the PRSPCODE table and creates one W3 method. To make it easier for the merchant to change the shipping method, the program looks for the Weight_Based shipping method and changes it. Therefore, there must be a shipping method called Weight_Based (see Figure 112 on page 163) before you enable the shipping by weight. The merchant can then switch the shipping code between W3 and Q3 for the weight-based shipping method and, thereby, enable or disable the shipping by weight.

The Ship cost by weight menu allows you to enter one price for each weight range for all destinations in the shipping price grid. An example for Alabama U.S.A. is given in Figure 133 on page 194.

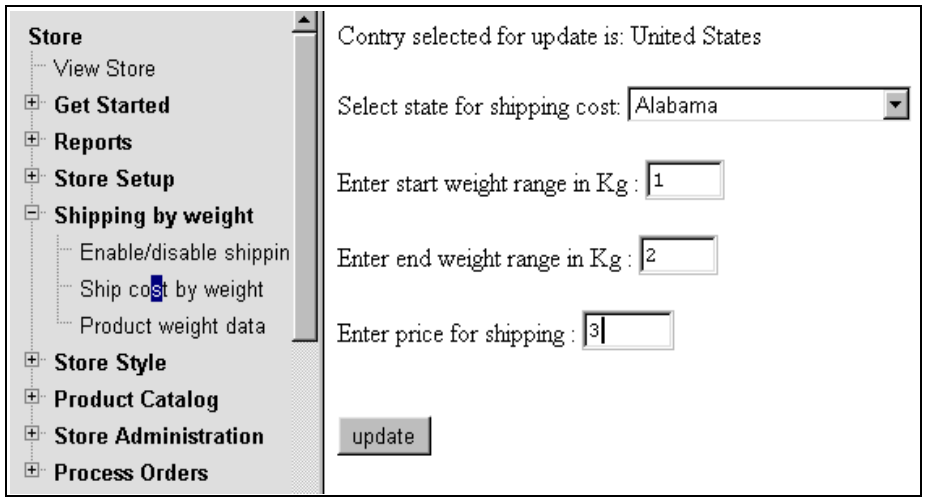


Figure 133. Entering shipping price data

The third menu item, Product weight data, is used to update the PRODUCT table with weight information. Weight data must be entered for each product that should be shipped by weight. Entering data will deposit the information into the PRODUCT table and change the pointer in the PRPSNBR column to point to the weight-based shipping method.

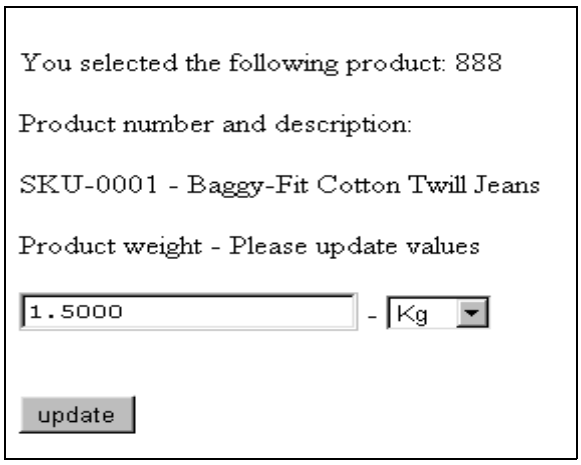


Figure 134. Entering product weight data

Now, when the system to update the information for shipping is ready, we can proceed and create the calculation process where we calculate the shipping cost based upon the entered information.

9.1.5 Calculating shipping cost

The program to calculate the shipping cost makes use of the tables ORDERS, SHADDR, SHIPTO, PRODUCT, PRSPCODE, and SHIPPING as mentioned in the previous section.
Shipping by weight can get very complicated and can require very detailed data entered into the system; this is not desirable in our environment where we want the data entered by merchants with limited time and resources. We must, therefore, make some assumptions to simplify the shipping by weight method:

- A shipping method, called Weight_Based, must be defined (see Figure 112 on page 163).
- The shipping cost for the maximum possible weight must be defined (see Figure 133 on page 194). This must be done for each jurisdiction.
- If the weight for one or more order items in the order is undefined, the shipping cost will be calculated by quantum.
- There must be a price grid defined for jurisdiction *Other*; this will be the price for all weight-based shipment not otherwise priced.
- All items in an order are shipped to the same address.
- The weight for all items in an order is summed up, and the shipping cost is calculated from the total weight of the order.
- Only one weight unit is supported.

The steps involved to retrieve the shipping cost are described in the following pseudo code example. The real code, written in C++, is listed in Appendix E.4, “C++ function GetOrdByWt.cpp” on page 346. This code replaces the default Overridable Function, GetOrdShTot_1.0. This new function is called GetOrdByWt:

```
GET ORDER_REF_NR
GET ORDERS WHERE ORDERS.ORDER_REF_NR
GET SHIPTO WHERE SHIPTO.STRFNBR = ORDERS.ORMORDER
FOR ALL IN SHIPTO DO
GET PRODUCT WHERE PRODUCT.PRRFNBR=SHIPTO.STRFNBR2
DONE
GET SHADDR WHERE SHADDR.SARFNBR=SHIPTO.STANBR
GET PRSPCODE WHERE PRSPCODE.PRRFNBR=PRODUCT.PRPSNBR
IF PRSPCODE.PSSPMTHD of any order line = Q3 THEN
```

```
       SKIP AND USE STANDARD METHOD
ELSE
FOR ALL IN PRODUCT DO
    SUM PRWGHT
DONE
GET SHIPPING WHERE CONTRY AND JURISTICTION IS = SHADDR. AND
             WIEGHT = PRWGHT
RETURN SHIPPING.SPCHRGE
```

9.1.6 Activating shipping by weight for a merchant

The new shipping Overrideable Function must be defined for WCS before it can be activated. The steps that must be completed to create and define the new OF are:

1. Generate and compile the C++ code to implement the function
2. Generate a library libITSO.a and copy it to /usr/lpp/NetCommerce3/bin directory
3. Register the new OF with WCS
4. Activate the new OF for our shop.

The C++ code is described earlier and listed in Appendix E.4. A make file to compile the code is listed in Appendix E.5, “Makefile for GetOrdByWt.cpp” on page 349. To use the make utility, type *make all* on a command line, the make utility takes the makefile as instructions to generate the library file libITSO.a, after the file is generated you must copy it to the /usr/lpp/NetCommerce3/bin directory manually.

Then to register the new OF *GetOrdByWt* in the WCS OF database, some SQL statements must be performed. The statements are listed in a SQL script in Appendix E.6, “reg_GetOrdByW.db2.sql SQL script” on page 350 to execute the script perform the following steps:

1. Log in as the DB2 instance owner (defaults to db2inst1).
2. Connect to the database by executing the following command:

   ```
   db2 connect to itso
   ```

   `itso` is the name of our database.
3. Run the script to register the function:

   ```
   db2 -tvf reg_GetOrdByW.db2.sql
   ```
4. Restart WCS

When the new OF is registered, you can assign it to a task using the NCADMIN interface of Net.Commerce.

1. Log on as administrator on NCADMIN with the following command:

   `http://HOSTNAME/ncadmin/`

2. Select **SITE MANAGER** in the left menu
3. Select **TASK MANAGEMENT** `in the` left menu
4. In the Select Task Type pull-down menu, chose **PROCESS**.
5. In the Task Name list, select **GET_ORD_SH_TOT**.
6. Select **TASK ASSIGNMENT** in the left menu.
7. In the store list select the mall or a store, we chose Dress demo in this example.
8. In the Overridable Functions box, select **GetOrdByWt(WCS SPE)(IBM ITSO)(1.00)**.
9. Click the **UPDATE** button to make the changes.

The ncadmin interface screen in Figure 135 shows the OF assignment. Since only the ASP has access to the ncadmin interface, this is a way for the ASP to control which merchants can make use of the shipping by weight function.

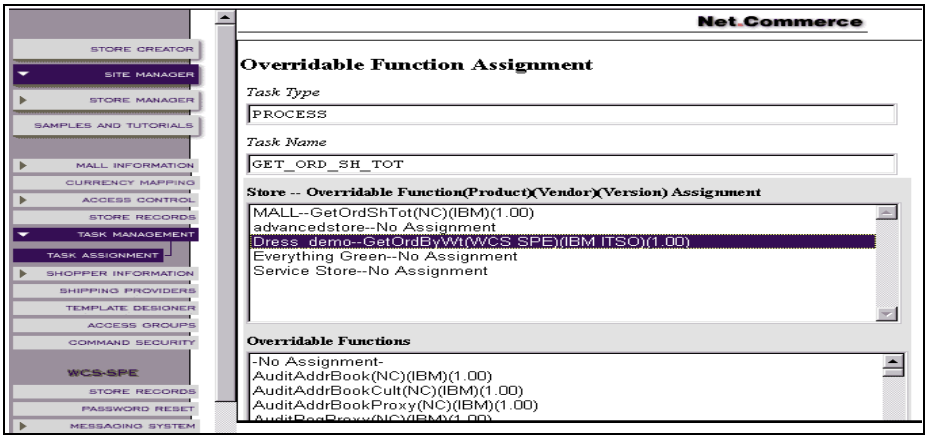


Figure 135. Enabling the cutomized shipping calculation

When the function is activated, the merchant can charging shipping costs by weight if weight data is entered into the system as described in Figure 131 on page 190 through Figure 134 on page 194.

9.2 Adding off-line payment methods

By default, the WCS has only one off-line payment method to offer merchants and customers, but, sometimes, it is necessary to have more than one method. By adding additional off-line payment options, the merchants can select from a list of offline payment methods, such as check, money order, or C.O.D. If there were three additional payment methods, the merchant would have to provide the customer with instructions in a message box. The order summary page should include the message and the order type.

This is important for ASPs in those countries that do not have good credit card regulations or for those merchants who want to sell with the C.O.D. method in the first transaction.

9.2.1 Payment methods in detail

When a merchant creates a new store or sets up the store, the merchant tool shows a screen like that shown in Figure 136.

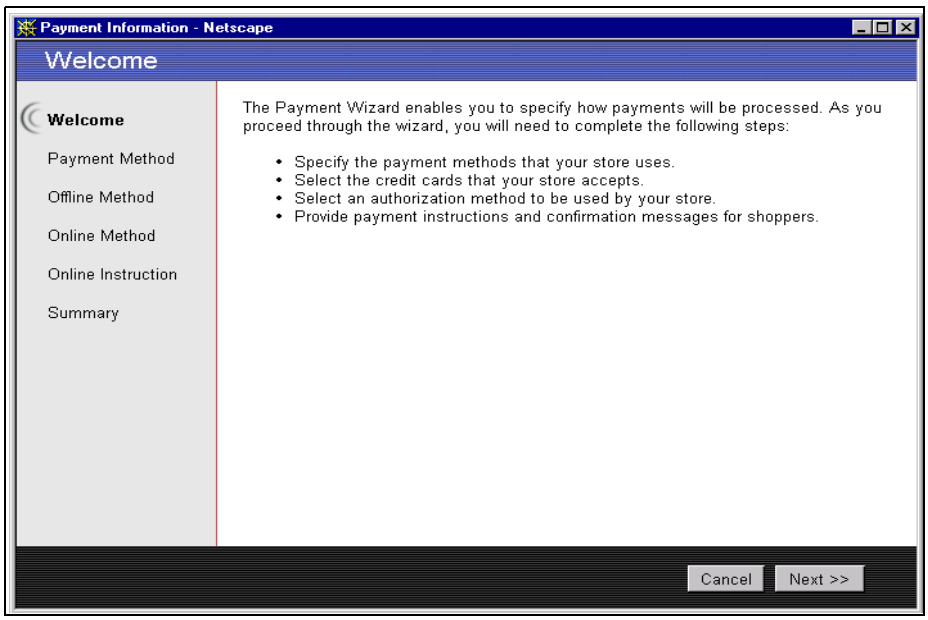


Figure 136. The welcome page of the payment setup option

The payment wizard contains a number of panels that, for example, offer alternative payment methods from which the merchant chooses to determine the payment method for the customer. These choices are stored by the

wizard and used to create the merchant's store when they press the Finish button at the end of the creation process. This information is held as an object called payment in the store creation wizard's client browser. Each panel adds more values to this object. When the Finish button is pressed on the final screen, a Java script function, convertToXML(), is called and converts this object into a number of XML parameters that are then sent to the servlet, MerchantAdmin.

This servlet does all the work of creating or modifying the new store based on the values in the XML parameters.

The PaymentWizard.xml file, located in
X:\Ibm\NetCommerce3\Tools\xml\nchs\payments (for Windows NT 4.0)
or
/usr/lpp/NetCommerce3/Tools/xml/nchs/payments (for AIX), is listed in Appendix F.1, "PaymentWizard.xml" on page 351.

As we can see in the file listing, there exists one panel name for each step in the payment wizard, and there is one payment method for each online payment method.

Because we are adding more options for the off-line payment methods, we must take a look at the offlineInstructionsTab panel name.

```
<panel name = "offlineInstructionsTab"
        url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.OfflineInstructions"
        helplink = "CTnchs.Payment.offlineInstructions.Help" />
```

Figure 137. The offlineInstructionsTab panel name

When the panel is executed, a call is made to the servlet, MerchantAdmin, to display the OfflineInstructions.tem template, which is shown in Figure 138 on page 200.

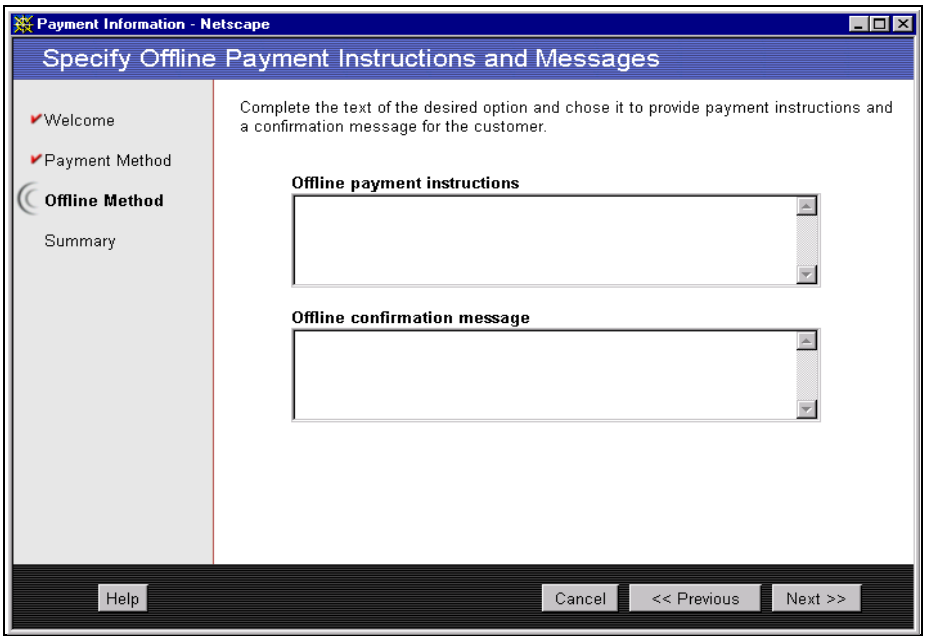


Figure 138. The off-line method screen

We can see that the merchant has only one option for off-line payments. Because we want to offer more options, it is necessary to customize the OfflineInstructions.tmp template, which is located in X:\Ibm\NetCommerce3\Tools\mpg_templates\nchs\payment (for Windows NT 4.0)
or
/usr/lpp/NetCommerce3/Tools/mpg_templates/nchs/payment (for AIX).

The next four figures show the original and changed templates; the changed templates are also listed in Appendix F, “Payment method code lists” on page 351.

```
Model nchs(paymentNLS)

display()
{
/*
<HTML>
<!--================================================================================
Licensed Materials - Property of IBM
```

Figure 139. OfflineInstructions.tmp template (1 of 2)

```
5648-B47

(C) Copyright IBM Corp.  1998, 1999, 2000.          All Rights Reserved

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
================================================================================-->
<LINK REL="stylesheet" HREF="/NCTools/html/common/pagestyle.css">
<SCRIPT SRC="/NCTools/javascript/Util.js"></SCRIPT>
<SCRIPT>
  var payment = top.get( "payment" );
  function initializeState()
  {
    document.f1.paymentInstructions.value =
    convertFromHTMLToText(payment.offlineInstructions);
    document.f1.confirmMessage.value =
    convertFromHTMLToText(payment.offlineConfirmation);
  }

  function validateEntries()
  {
    if ( document.f1.paymentInstructions.value.length > 256 ||
         document.f1.confirmMessage.value.length > 256 ) {
       alert ("$paymentNLS.offlineMsgTooLong$");
       return false;
    }
    payment.offlineInstructions =
    convertFromTextToHTML(document.f1.paymentInstructions.value);
    payment.offlineConfirmation =
    convertFromTextToHTML(document.f1.confirmMessage.value);
    top.put( "payment", payment );
    return true;
  }
</SCRIPT>
<BODY BGCOLOR="#CCCCCC" ONLOAD="initializeState();">

<FORM NAME="f1">
  $paymentNLS.offlineDesc$<BR><BR><BR>

  <UL TYPE="DISC">
    <B>$paymentNLS.offlineInstructions$</B>
    <BR>
    <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions" ROWS=4 COLS=35></TEXTAREA>
    <BR>
    <BR>
    <B>$paymentNLS.offlineConfirmation$</B>
    <BR>
    <TEXTAREA WRAP=PHYSICAL NAME="confirmMessage" ROWS=4 COLS=35></TEXTAREA>
  </UL>
</FORM>
</BODY>
</HTML>
*/
}
```

Figure 140. OfflineInstructions.tem template (2 of 2)

We can customize the template by adding some lines to it to offer more options, such as payment with check, C.O.D., and many more options. In the next two figures, we can see the new OfflineInstructions.tem template after the customization:

```
Model nchs(paymentNLS)

display()
{
/*
<!--========================================================================
Licensed Materials - Property of IBM
5648-B47
(c)  Copyright  IBM Corp.  1998, 1999.      All Rights Reserved
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
========================================================================-->
<HTML>
<LINK REL="stylesheet" HREF="/NCTools/html/common/pagestyle.css">
<SCRIPT>
  var payment = top.get( "payment" );
  function initializeState()
  {
    document.f1.paymentInstructions1.value = payment.offlineInstructions;
    document.f1.confirmMessage1.value = payment.offlineConfirmation;
  }
  function validateEntries()
  {
    if ( document.f1.paymentInstructions1.value.length > 256 ||
         document.f1.confirmMessage1.value.length > 256 ) {
       alert ("$paymentNLS.offlineMsgTooLong$");
       return false;
    }
    payment.offlineInstructions = document.f1.choseninstr.value;
    payment.offlineConfirmation = document.f1.confirmMessage1.value;
    top.put( "payment", payment );
    return true;
  }
</SCRIPT>
<BODY BGCOLOR="#CCCCCC" ONLOAD="initializeState();">
<FORM NAME="f1">
$paymentNLS.offlineDesc$<BR>
<UL TYPE="DISC">
<input type=radio name=choose
onclick="this.form.choseninstr.value=this.form.paymentInstructions1.value;" CHECKED>
    <B>$paymentNLS.offlineInstructions$</B>
    <BR>
```

Figure 141. The first part of the new OfflineInstructions.tem template file

```
 <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions1" ROWS=2 COLS=35>
Call us to a number 1-800-___-____ to bring your credit card information
and order number.</TEXTAREA>
<BR>
<BR>
<input type=radio name=choose
onclick="this.form.choseninstr.value=this.form.paymentInstructions2.value;">
    <B>Pay with check</B>
    <BR>
    <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions2" ROWS=2 COLS=35 >
Send us your check to the account number xxx-xxxxxxx of the xxxxxxxxxxx
bank to complete your order.</TEXTAREA>
<BR>
<BR>
<input type=radio name=choose
onclick="this.form.choseninstr.value=this.form.paymentInstructions3.value;">
    <B>Cash On Delivery (C.O.D.)</B>
    <BR>
    <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions3" ROWS=2 COLS=35 >
   Just pay your order when you recive it in your home. As easy as it!</TEXTAREA>
<input type=hidden name=choseninstr value="">
<BR>
<BR>
<BR>
<B>$paymentNLS.offlineConfirmation$</B>
<BR>
<TEXTAREA WRAP=PHYSICAL NAME="confirmMessage1" ROWS=1 COLS=35>We appreciate a lot
your purchase. For any information about your order, please see the Customer
Information Section.</TEXTAREA>
  </UL>
</FORM>
</BODY>
</HTML>
*/
}
```

Figure 142. The last part of the new OfflineInstructions.tem template file

To see these new changes, we must stop and restart the Websphere Servlet Service, and, finally, we can see a screen like the one shown in Figure 143 on page 205.

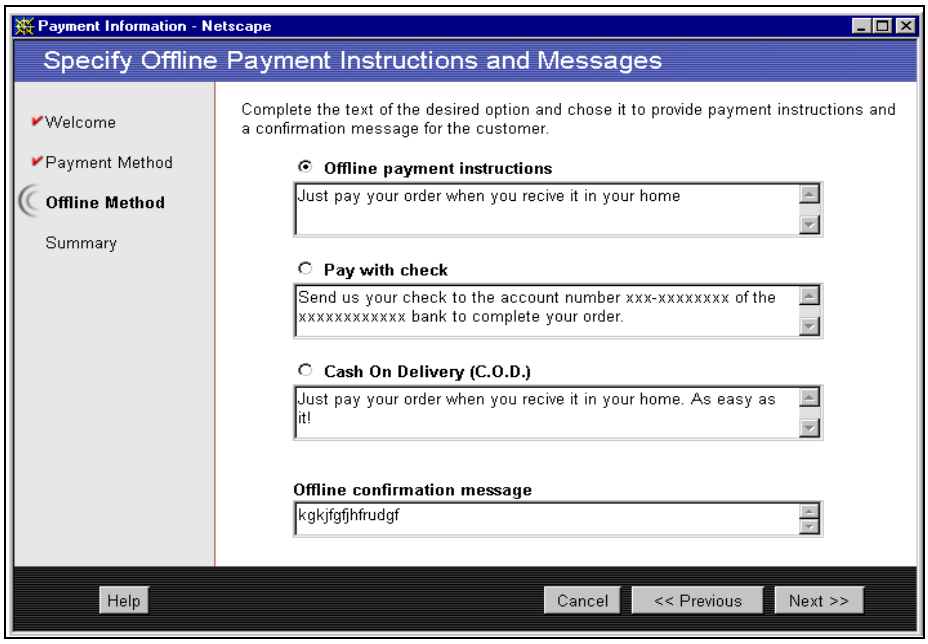


Figure 143. The off-line method screen after customization

9.3 Gift messages/wrapping

In this section, we will go into detail about how to implement gift messages and gift wrapping as new features in the WCS SPE environment. We will create the example and explain in detail how to expand and use the existing database model to store information specific to this new feature, and we will show how to enable the feature for a single store only.

Adding and implementing this new feature will be an example of an advanced customization that requires knowledge about the WCS framework as well as a good understanding of how Net.Commerce works.

In the example, we will be modifying a store created as an advanced store using the classic store model.

Note

It is very important to note that the Net.Data macro should not be used for updating the database in the Net.Commerce system. We have done it here to make the interface simple. It is always recommended to use commands and Overridable Functions to achieve any update to the database in your implementation.

9.3.1 The scenario

To illustrate the process of adding a new feature to WCS SPE, consider the following scenario: The ISP has purchased WCS SPE and is already hosting many merchants. However, some of the merchants are asking for a special feature to be used when shipping gifts for special occasions, such as Christmas, Easter, or birthdays. The merchants want to be able to offer their customer the opportunity to personalize a gift message and, optionally, select a special gift wrapping when shipping their purchase as part of a gift. Because only some of the merchants might have the business model where gift messages and wrapping are appropriate, this service should be offered as an option to the merchants. Also, not all merchants might want this feature to be enabled all the time but only on special occasions, which means that the merchant must be able to enable/disable the feature at any time they desire.

9.3.2 Planning the feature

Adding a feature, such as gift message/wrapping, requires some modification, which means the feature must be carefully specified. Also, since adding this type of feature requires restarting WCS, you should inform your merchants about their services being down for a short period of time, for example, with service window.

When designing this feature, the following things must be considered before implementation:

- The ISP must be able to offer the feature to some merchants.
- The merchants must be able to enable and disable the feature.
- If the merchant has enabled the feature, it should appear to the shopper during the shopping flow.
- The merchant might want to present information on how to use the feature; so, each merchant must be able to define their own help text.

- If the shopper has entered a message and/or selected a gift wrapping, it must be stored somewhere so the merchant can see this information.

The implementation of the listed items will require a lot of changes to be made. Besides implementing the actual logic for the features, we also need to customize some Net.Data macros to include the gift feature in the shopping flow, and we need to make use of the database model to be able to store information, such as the actual message, the gift wrapping selected, and whether the feature is enabled for this merchant.

9.3.3 Designing the feature

With the specifications, we now need to design the actual solutions, which means that we will have to specify which parts need to be done. For this feature, parts are visible to the ISP and the merchant, and some parts are visible to the shopper.

Options visible to the ISP

The ISP must be able to enable and disable this feature to each merchant, this could be part of charging the merchant for using the feature. The easiest way to do this is to have the information available in the NCADMIN interface, which the ISP will use.

Options visible to the merchant

When necessary, the merchant should be able to enable and disable and change the text that will appear as part of the feature. For administrative purposes, the merchant should also be able to access the choices made by the shopper, which means the actual message and the wrapping the shopper selected. We decided to show this information as part of the *manage order* page in the merchant tool and add a new page to the tool where the merchant can enable and disable the feature and change the text explaining how to use the feature.

Options visible to the shopper

If the feature is enabled for a particular store, the shopper must be presented with a page where it is possible to type in a gift message and select one of the gift wrappings, together with instructions on how to use the feature.

We think the most natural place to display this information and collect the input would be during the checkout process. We then decided to expand the “Payment information” page with a section including the gift message/wrapping feature. For this purpose, we will have to change the macro, ord_pay.d2w, to include a function that checks whether this feature is enabled for the store, and, if so, it will add some HTML to the payment page.

9.3.4 Database model

As can be seen from the description of the feature, we need to be able to store some information when using it. We will have to explore the Net.Commerce database model, since this is the obvious place to store this kind of information.

To store new data in Net.Commerce, you got the these possibilities:

1. Add new fields to existing tables. For example, some of the data we need to store is related to the merchant, and in the tables, MERCHANT and MCSPINFO, there is already one row for each merchant. Adding new fields to one of these tables is quite easy but has a huge drawback: Since we are actually making changes to the database scheme, we will, most likely, run into problems if we later decide to upgrade your server to the next version of WCS SPE.
2. Use *reserved for customization* fields. Many tables in the WCS database scheme contain fields that can be used for customization. The MERCHANT table, for example, contains two custom fields, each having the datatype, varchar(254). Using custom fields is nice if you need to supply additional information, but, since there are only a limited number of custom fields in each table, you might run out of unused fields as you start to extend WCS.
3. Create new tables with the columns needed. This will separate the feature from the WCS system but will require us to maintain the relationship to the database. We also need to take care of cleaning up unused data. However, by using the existing referential integrity, this could be a fairly easy task.

In this example, we will use options two and three. Option one is not the recommended way to store additional informations; so, we will just show how to add a new table and how to use the custom fields in existing tables.

For each merchant, we need to store the following information:

Table 18. Additional table for gift message/wrapping

Column name	Type	Description
GMMENBR	INTEGER	Merchant reference number. This is a foreign key that reference column MERFNBR in table MERCHANT

Column name	Type	Description
GMALLOW	SMALLINT	0 - Merchant is not allowed to use this feature 1 - Merchant is allowed to use this feature
GMENABLED	SMALLINT	0 - Feature disabled in store 1 - Feature enabled in store
GMDESC	VARCHAR(254)	The information text to be shown at the gift page

To create the database table shown in Table 18, perform the following steps:

1. Log in as the DB2 instance owner (defaults to user db2inst1).
2. Create a file, called giftmessage.db2.sql, which will contain the SQL.

   Add the following code to the script file:

```
create table giftmessage (
        gmmenbr integer not null,
        gmallow smallint,
        gmenabled smallint,
        gmdesc  varchar(254),
        constraint p_giftmessage primary key (gmmenbr),
        constraint fme_giftmessage foreign key (gmmenbr)
                references merchant (merfnbr)
                on delete cascade
);
```

3. Save the file and exit editor.
4. Connect to the database by executing the following command:

```
db2 connect to itso
```

   In our example, the database is called ITSO.

5. Run the following script to create the table:

```
db2 -tvf giftmessage.db2.sql
```

This is the SQL that will generate the table, and the output should look something like that shown in Figure 144 on page 210.

```
f50::db2inst1:/home/db2inst1/GiftMessage>db2 connect to itso

   Database Connection Information

 Database server        = DB2/6000 6.1.0
 SQL authorization ID   = DB2INST1
 Local database alias   = ITSO

f50::db2inst1:/home/db2inst1/GiftMessage>db2 -tvf giftmessage.db2.sql
create table giftmessage ( gmmenbr              integer not null, gmallow
        smallint, gmenabled     smallint, gmdesc                varchar(254),
constraint p_giftmessage primary key (gmmenbr), constraint fme_giftmessage
foreign key (gmmenbr) references merchant (merfnbr) on delete cascade
DB20000I  The SQL command completed successfully.

f50::db2inst1:/home/db2inst1/GiftMessage>
```

Figure 144. Output from creating a new table

Now, the table is created and we can continue developing the macros that will control which merchants can use this feature.

9.3.5 Implementing the ISP interface

Now that we have created the new database table, we can start implementing the tool that the ISP will use to enable the feature for stores.

The ISP will be presented with an HTML page from which they can retrieve information about the status of each merchant. When a merchant is selected, a macro page will show the current gift message/wrapping information for the selected merchant. The ISP can then choose to update the availability of the feature or select a new merchant.

To keep things simple, we have not integrated the new macros into the NCADMIN interface of Net.Commerce. The interface itself is a frame structure in which the upper frame shows a pull-down menu containing all the merchants. When a merchant is selected, the required information is retrieved from the database and shown in the bottom frame.

Figure 145 on page 211 shows what the interface looks like when a store has been selected and the information displayed to the ISP.

Figure 145. The ISP interface for gift message/wrapping

The source code for the HTML and macro files can be found in Appendix G, “Source code for gift message/wrapping” on page 357. The file, giftadmin.htm, should be put in the following directory:

/usr/lpp/NetCommerce3/html/en_US/ncadmin/sitemgr/

and the two macro files should be put in:

/usr/lpp/NetCommerce3/macro/en_US/ncadmin/sitemgr/

To launch the ISP interface in your browser, type the following URL in your browsers location field:

```
http://HOSTNAME/ncadmin/sitemgr/giftadmin.htm
```

Before you launch the URL, make sure you are already logged on as an administrator; otherwise, you will be presented with the mall logon page.

9.3.6 Changing the store page

We need to modify the macro ord_pay.d2w so it displays the gift message fields when the feature has been enabled by the ISP and the merchant.

To make this new feature available to all merchants, we need to change the default ord_pay.d2w macro for the advanced store, which can be found in the following directory:

/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/

It is also possible to customize the macro individually for each merchant, but, for now, we will just change the default mall macro to include the functionality to handle the gift message part.

We will modify the ord_pay.d2w macro by adding a function that will check whether the gift message feature is enabled in the shop. The function queries the database against the table we created to see if the ISP and the merchant have enabled the feature. As a result, the function sets some variables so that, later in the HTML section, we can decide whether or not we should add the extra HTML for the feature.

The function that we have added looks something like the following:

```
%{ Function to retrieve gift message informations about current merchant
    if the feature is enabled and the merchant is allowed to use it. %}
%FUNCTION(dtw_odbc) GetGiftMessageInfo(){
    SELECT
        gmdesc
    FROM
        giftmessage
    WHERE
        gmmenbr = $(MerchantRefNum) AND
        gmallow = 1 AND
        gmenabled = 1

    %REPORT{
        %ROW{
            @DTW_assign(GIFTMSGENABLED, "yes")
            @DTW_assign(GIFTMSGTEXT, V_gmdesc)
        %}
    %}

    %MESSAGE{
        default : { @DTW_assign(GIFTMSGENABLED, "no") %} : continue
    %}

%}
```

In the HTML section of the macro, we will then call the GetGiftMsgInfo() function; then, we can check the GIFTMSGENABLED variable to see whether or not we will have to show the extra HTML.

If the feature is enabled, the HTML we add will print out the information text, which can be fetched from the GIFTMSGTEXT Net.Data variable and add two extra input fields to the order form.

The first field is an input field of type text, which we will name giftmsg, where the customer can type in the message they want. The second type of field is the four radio buttons, named wrapping, where the customer can choose one of the four gift wrappings. The number of gift wrappings is fixed and requires the merchant to have uploaded four images. In our example, we made four images, each 48x48 pixels, and placed them in the following directory:

/usr/lpp/internet/server_root/pub/<storename>/images

The images should be named giftwrap1.gif, giftwrap2.gif, giftwrap3.gif and giftwrap4.gif. If you want to change the file names, you must change them in the ord_pay.d2w macro. The <storename> can be fetched from the LongStoreName variable, as can be seen in the example code.

Our extension to the HTML section looks like this:

```
@GetGiftMessageInfo()
%IF(GIFTMSGENABLED == "yes")
            <TABLE BORDER="1" WIDTH="100%" CELLPADDING="0"
CELLSPACING="0">
            <TR>
               <TD><TABLE BORDER="0" WIDTH="100%" CELLPADDING="0"
CELLSPACING="4">
                  <TR VALIGN="TOP">
                     <TD WIDTH="50%"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(GIFTMSGTEXT)</FONT>
                     </TD>
                     <TD VALIGN="MIDDLE" ROWSPAN="2" WIDTH="50%"><CENTER>
                     <INPUT TYPE="radio" name="wrapping" value="1"><IMG
SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap1.gif" BORDER="0"> 
                     <INPUT TYPE="radio" name="wrapping" value="2"><IMG
SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap2.gif" BORDER="0">
                     <BR>
                     <INPUT TYPE="radio" name="wrapping" value="3"><IMG
SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap3.gif" BORDER="0"> 
                     <INPUT TYPE="radio" name="wrapping" value="4"><IMG
SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap4.gif" BORDER="0">
                     </CENTER></TD>
```

```
                    </TR>
                    <TR>
                       <TD WIDTH="50%"><INPUT TYPE="text" NAME="giftmsg"
SIZE="24" MAXLENGTH="254">
                       </TD>
                    </TR>
                    </TABLE>
                 </TD>
              </TR>
           </TABLE>
%ENDIF
```

We have placed the extra HTML code just before the submit button to be sure it is with the FORM tag of the payment form.

In Figure 146, you can see the output from the new ord_pay.d2w macro when the gift message/wrapping feature is enabled.

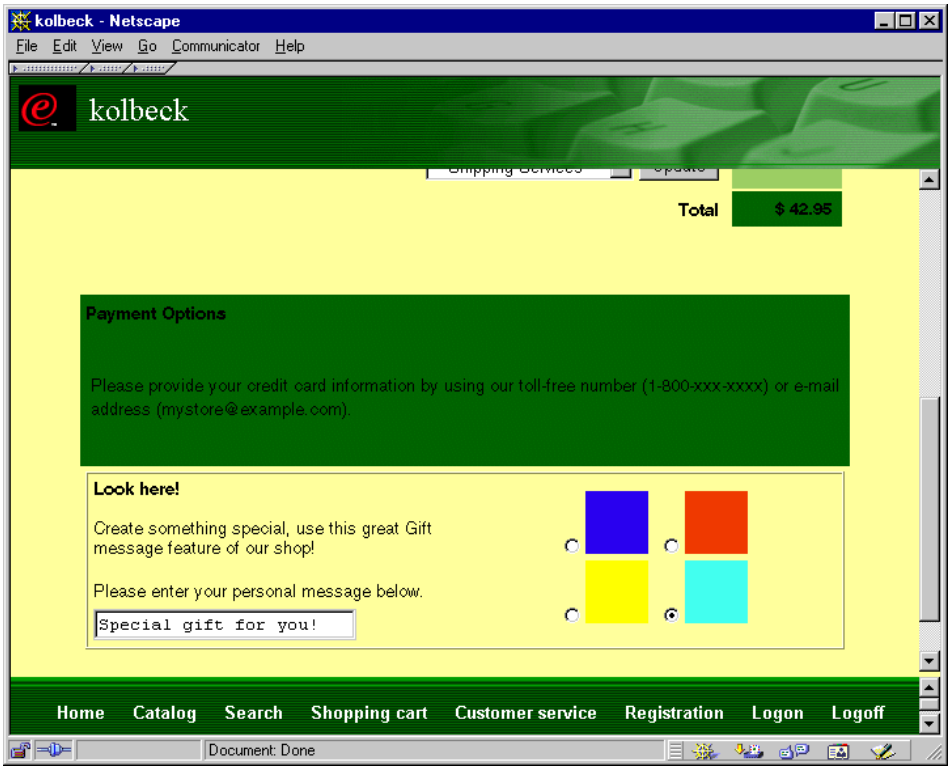


Figure 146. The Payment page with gift message feature enabled

9.3.7 Implementing the business logic

The modification we made to the ord_pay.d2w macro included two new fields in the order form. However, these elements are HTML, which means that the Net.Commerce command, `OrderProcess`, which will process the order, does not know anything about them. To be able to handle the extra information that the ord_pay.d2w macro might send and to store this information together with the current order, we need to extend the OrderProcess command.

The manual, *Commands, Tasks, Overridable Functions and Database Tables*, which is available at the Web site,
`http://www-4.ibm.com/software/webservers/commerce/servers/lit-tech-general.html`, and describes how all the commands in WCS work. From the description of the command, we can see that the `OrderProcess` command calls a task with the name, EXT_ORD_PROC, to perform additional processing that is needed.

From the description of the EXT_ORD_PROC task, we see what input parameters we can expect, how we should handle errors, and that this task is, by default, assigned to the OF DoNothingNoArgs(). As the name of the OF says, the default function does nothing.

What we have do to be able to process the extra information is to write an OF and assign it to the EXT_ORD_PROC task, which is then executed by the `OrderProcess` command. The new OF, which we will call addGiftMsg, takes the gift message and the selected wrapping from the HTML form and stores the information with the current order. From the description of the database tables, we can see that the ORDERS table contains extra fields for customization. We will use two of the fields. In ORFIELD1, we will store the number of the selected wrapping, and, in ORFIELD3, we will store the message.

If you want to get into details about how to write commands and Overridable Functions for Net.Commerce, it is worth taking a look at the book *Commands, Tasks, Overridable Functions and the E-Commerce Data Objects*, which can be found on the Net.Commerce Web site:
`http://www.software.ibm.com/commerce/net.commerce`

This book describes in detail the programming framework of Net.Commerce v3.x and is a must for anyone who is interested in customizing and extending Net.Commerce.

In this example, when you compile the code found in Appendix G.6, “Source code for addGiftMsg.cpp” on page 363, using the makefile, you will end up with a library, named libITSO.a, which you must copy to the

/usr/lpp/NetCommerce3/bin directory so that Net.Commerce can access our new OF. Before you can assign the EXT_ORD_PROC task to our new function, the addGiftMsg function must be registered in the database. We will create an SQL script to register the function as we did when we created the new table in the database:

1. Log in as the DB2 instance owner (defaults to db2inst1).
2. Create a file, called reg_giftmsg.db2.sql, and type in the following SQL code:

```
-- *************************************************************
-- * Register of addGiftMsg for PROCESS TASK EXT_ORD_PROC
-- *
insert into ofs(refnum, dll_name, vendor, product, name, version)
values(
(select max(refnum)+1 from ofs),
'libITSO.a',
'IBM ITSO','WCS SPE',
'addGiftMsg',1.0
);
```

3. Save the file and exit the editor.
4. Connect to the database by executing the following command:

   ```
   db2 connect to itso
   ```

   `itso` is the name of our database.
5. Run the script to register the function:

   ```
   db2 -tvf reg_giftmsg.db2.sql
   ```
6. Restart WCS

When the new OF is registered, you can assign it to the task using the NCADMIN interface of Net.Commerce.

1. Log on as administrator on NCADMIN with the following command:

   ```
   http://HOSTNAME/ncadmin/
   ```
2. Select **SITE MANAGER** in the left menu
3. Select **TASK MANAGEMENT** `in the` left menu
4. In the Select Task Type pull-down menu, chose **PROCESS**.
5. In the Task Name list, select **EXT_ORD_PROC**.
6. Select **TASK ASSIGNMENT** in the left menu.
7. In the Overridable Functions box, select **addGiftMsg(WCS SPE)(IBM ITSO)(1.00)**.

8. Click the **UPDATE** button to make the changes.

Figure 147 shows the screen just before pressing the UPDATE button, and, as you can see, the task was, by default, assigned to the DoNothingNoArgs function.

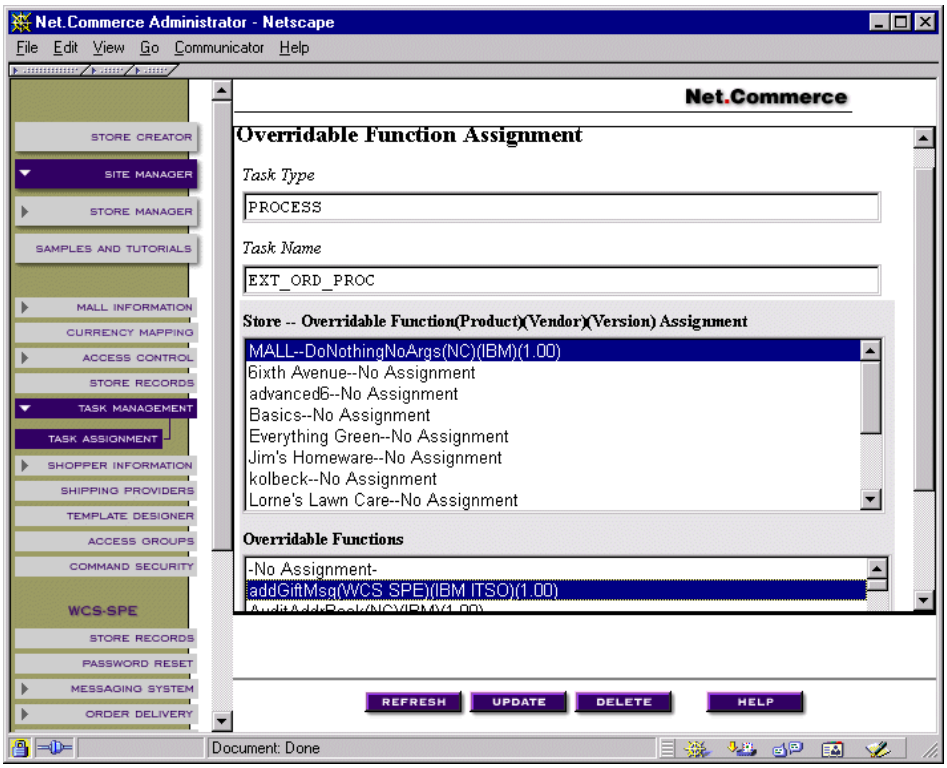


Figure 147. Assign addGiftMsg Overridable Function

9.3.8 Extending the merchant tool

The merchant needs to be able to change the availability of the feature as well as change the text that will be shown on the payment page. The merchant interface to the gift message/wrapping is, basically, the same as the ASP interface, except that the merchant can also change the text message.

Since the merchant will be using the merchant tool, we will have to add an extra feature to the menu of the tool, which will link to our macro that will show the current status of the feature for the store. On this page, the merchant can enable or disable as well as change the info text.

The macro is named giftadminstore.d2w and is placed in the /usr/lpp/NetCommerce3/macro/en_US/ncadmin/storemgr/ directory.

The source code can be found in Appendix G.4, “Macro file giftadminstore.d2w” on page 360.

To add this macro to the merchant tool for the advanced store under *Store Setup*, do the following:

1. Locate and open the xml file, navigation.xml, for the navigation bar in the advanced store, which is found in /usr/lpp/NetCommerce3/Tools/xml/nchs/mtool/advanced/
2. In the file, locate the line that reads <node name="storeSetup", and, just before the closing tag, </node>, add the following new link tag:

   ```
   <node name="adminGift"
   url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/giftadminstore.d2w/
   report"
               image="/NCTools/images/clear.gif"
               users="siteAdmin storeAdmin" />
   ```

   where:

   - `name` is the variable used to identify the link.
   - `url` is the action of the link.
   - `images` defines the image on the navigation bar to the left of the link.
   - `users` defines which users are allowed to see this link.
3. Save the XML file.
4. Locate and open the file, navigationNLS.properties, which, at the time of this writing, can be found in /usr/lpp/NetCommerce3/Tools/lib/nctools.jar, and add the following line:

   ```
   adminGift=Gift Messages
   ```
5. Save the file, re-create the nctools.jar, and restart WCS.

After restarting WCS, you can launch the merchant tool to enable the feature for a particular store.

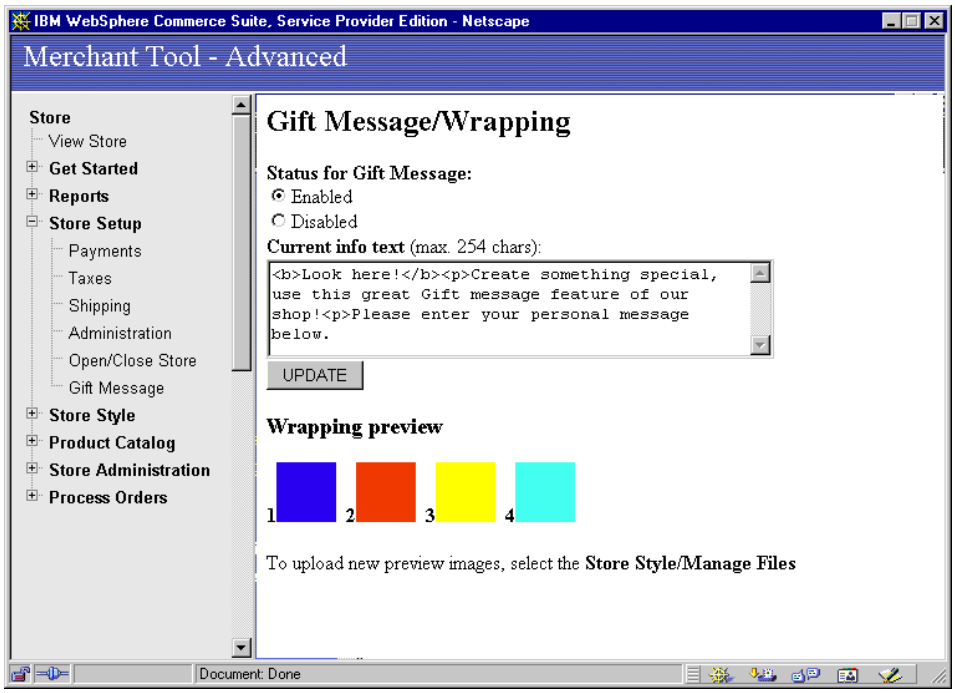


Figure 148. The merchant tool with gift message

When the merchant receives the order, he or she must be able to see whether the customer has entered a gift message or selected a gift wrapping. We will extend the Order Details page in the merchant tool to include information given by the customer during the purchase. For more details about the template files in WCS SPE, read the manual, *Customization Guide, Version 3.2*, which is shipped with the product.

The template file for the order details page, which we will change, is named OrderDetails.tem, and is located in the following directory:

/usr/lpp/NetCommerce3/Tools/mpg_templates/nchs/order_mgmt/

and we will extend it with a function to retrieve and display the information entered by the customer at the payment page. Perform the following steps:

1. Open the file in your editor.
2. Find the line where the addComments() function is called. The lines should look like the following:

```
--
-- Add the order comments
```

```
--
/* <BR><BR> */
horizontalSeparator(orderMgmtNLS.orderDetailsCommentSeparator)
addComments()
```

3. Insert a call to our new function that will display gift message information by adding the following line:

```
--
-- Show gift message/wrapping informations
--
addGiftMessage()
```

4. Add the following function to the file:

```
--
-- Add gift message/wrappings for this order
--
addGiftMessage()
{
   Query stmtgift
   Var   giftmsg
   Var   wrapping

   giftmsg = ""
   wrapping = ""

   stmtgift = "SELECT ORFIELD1, ORFIELD3 FROM ORDERS WHERE ORRFNBR=" +
parameters.selectedOrders

   stmtgift|reset()
   wrapping += stmtgift.orfield1
   giftmsg += stmtgift.orfield3

   if ( giftmsg != "null" ) {
      /*
      <H3>Gift Message:</H3>
      $giftmsg$
      */
   }
   if ( wrapping != "0" && wrapping != "null" ) {
      /*
      <H3>Selected Wrapping</H3>
      Wrapping no. $wrapping$
      */
   }
}
```

We added the code just before the addComment() function in the template file.

5. Save the template file and restart WCS.

After the modification of the Order Details page, the merchant will be able to see the information that the customer has entered during the checkout, which is shown in Figure 149. The modified OrderDetails.tem file is also shown in its complete form in Appendix G.7, “Modified OrderDetails.tem template” on page 366.

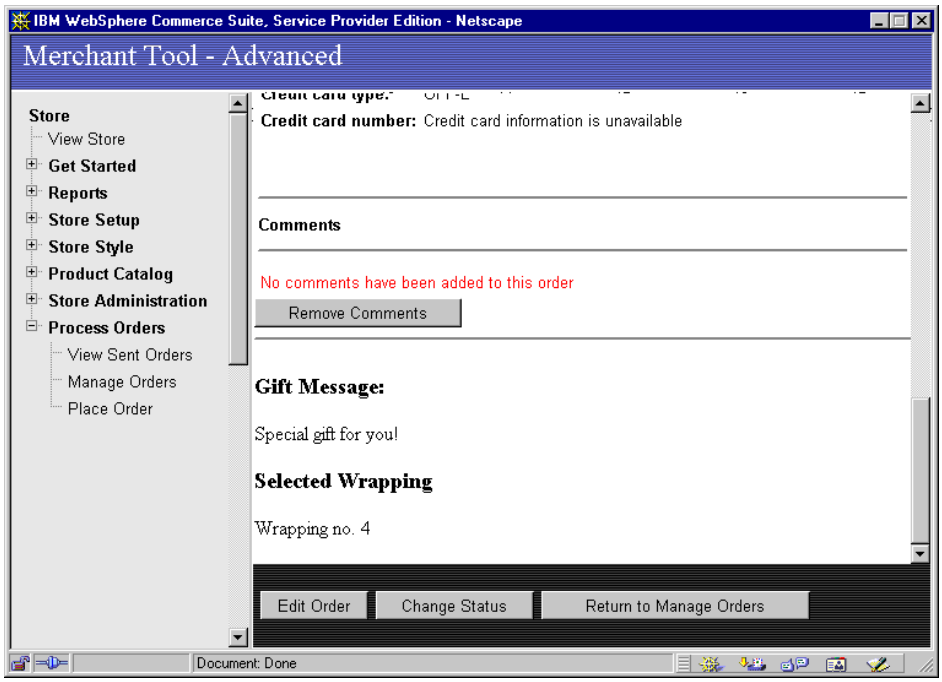


Figure 149. Order Details page in the merchant tool

Now that we have implemented everything, it is possible to test the overall functionality of the gift message/wrapping feature. Using the URL

`http://HOSTNAME/ncadmin/sitemgr/giftadmin.htm`

you should be able to enable or disable the new feature for each of your merchants as an ASP. This page, intended for the ASP only, makes it easy to handle the accounting as you can see if the feature is enabled in one store or not. If a store wishes to use the feature, the ASP can enable it in that particular store and then add the use of it to the bill.

As a merchant, you can use the merchant tool to enable or disable the feature in your store or change the information text that is displayed on the payment information page. You can also test your new payment page, place a test order that includes a gift message, and then retrieve this information on the Order Details page in the merchant tool.

9.3.9 How to charge the customer

Until now, using the feature in the shop has been free of charge for the shopper. However, since it might require some extra work for the merchant to handle the request about gift messages, sometimes, the merchant might want to charge the shopper for using the feature. Since the feature is implemented in this chapter, the following possibilities for charging the shopper exist.

Make a note and change the order in the merchant tool
When the shopper is on the payment page, you should make a note saying something like the following: *Note: Using the gift message 1$ will added when your order is processed.* However, the shopper will not see this extra charge as part of the payment information, which is the bad side of this approach, since he or she will not immediately see how much the total will be. In some countries, there might also be some legal issues about adding charges after the shopper has accepted the purchase.

Charging for the use of the feature can be done by the merchant using the merchant tool. When the merchant is processing the orders, he or she can chose to edit the orders that make use of the gift message and then add the charges to those orders.

Add a special shipping provider
The charge of using the gift message feature can also be done by adding an additional shipping service called, for example, gifts, which then adds the amount to the purchase when the shopper has selected to use the gift message. In order to implement this, you will need to change the orderpay.d2w macro further so that it uses the standard shipping by default. But, when the shopper decides to use the gift message feature, the page should recalculate the shipping, now including the use of the feature.

This implementation works best if the merchant only has one shipping provider; otherwise, they will have to define the shipping service with gift messages for each of their shipping providers. However, having the orderpay.d2w macro recalculate the order makes it possible for the shopper to see the exact amount just before finalizing the order.

Add a virtual product called gift message

The default shop a merchant gets when he or she creates a store makes it possible for the shopper to add a comment to every item in the shopping basket.

If you add a product, called Gift Message, to your catalog, the shopper can then type in the message in the comment field if he or she decides to add the product to the shopping card. The product should be uncategorized so that it doesn't show up on the product catalog. Even if the product is uncategorized and, thus, is not shown when browsing the catalog, the shopper will still be able to find the product if he or she searches for it.

On the Payment Information page, one could place a button that reads *Add gift message*, which then adds this product to the shopping card and shows the shopping basket where the shopper can type in the message.

This implementation also makes it possible to calculate the exact amount that the shopper has to pay. However, even this implementation is quite easy for the merchant; it gives no way for the ISP to control which merchant uses a special feature like this, and, thus, the ISP cannot charge the merchant for using such a feature.

Chapter 10. Provisioning

Many ISPs already have an existing site offering various services to the customer. This chapter discusses how to simply extend that existing site to include WCS SPE. First, we will look at the direct URLs to call when creating a basic or advanced store. This is done instead of going through the standard WCS site page. Finally, we will use a common scenario where an ISP has one custom registration page for all of their services and explain how to pass this information to the store creation wizard so the customer does not need to retype it.

10.1 Calling the store creation wizard directly by URL

We will outline the URLs to call to create the basic or advanced stores. To let the ISP include the URLs on previously-created pages.

10.1.1 Basic store

The URL to call the create basic store wizard is as follows:

```
http://<hostname>/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateS
tore&storeXML=nchs.store_creator.basic.xml&level=Basic&catalogXML=nchs.cat
alog.simpleCatalog.xml&bannerHTML=registerBannerBasic.html
```

This will call the first page in the creation process, but it will appear in the same browser window from which it was called. The default installation of WCS displays the store creation wizard in a new browser window. If you want it called in the same browser window, you will have to change the action of the Close button on the final wizard screen. It expects to be in a new window and attempts to close it. The template containing the close button is the file:

/usr/lpp/NetCommerce3/Tools/mpg_template/nchs/store_creator/NavClose.tem

It is necessary to change the JavaScript function, closeMe(), in that file as shown in Figure 150 on page 226. Remove the text marked in italics and insert the text marked in bold. For the changes to take effect, WebSphere must be restarted.

```
<SCRIPT>
     function closeMe()
     {
       var cts = new Date();

*/
       Var serviceDataHome = homeDirectory.lookup("ServiceDataHome")
       Var serviceDataFilter = serviceDataHome.createFilter()

       serviceDataFilter.setLevelEquals(merchant.level)
       serviceDataFilter.setAttributeEquals('mtoolXML')

       Var mtoolXML = serviceDataHome.find(serviceDataFilter);
/*
      var cts = new Date();
      var loc = 'http://$env.hostname$/servlet/MerchantAdmin?'
                  + 'DISPLAY=CTnchs.mtool.MerchantTool&CTS=' + cts.getTime();

   // remove all commented out below
    /* var newwindow =
top.getPanelAttribute("storeCreatorPanelVerify","mtoolInNewWindow");
       if (newwindow == "NO" && top.window.opener.closed == false ) {
         top.location.href = loc;
       } else {
         window.open(loc);
         top.close();
       }
    */
   // And add the following line
   top.location.href = loc;
     }
   </SCRIPT>
```

Figure 150. CloseMe function

If you want the store creation wizard to appear in a new window as in the WCS, you can add the JavaScript code, shown in Figure 151 on page 227, to your custom page and you then call the function with:

```
<a href="javascript:create_basic()">Click here to create a Basic Store</a>
```

You will notice that the URL for a basic store, called basic and marked in bold in Figure 151 on page 227, has an extra parameter, called CTS. This is set to the current time and is used to prevent caching of this page by the browser. It is optional (but recommended) to include this since browser caching can cause problems. The following code calls the basic store wizard with non-caching functionality added and with the CTS parameter set as a random number.

```
<a href="javascript:window.location =
'/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nc
hs.store_creator.basic.xml&level=Basic&catalogXML=nchs.catalog.simpleCatal
```

```
og.xml&bannerHTML=registerBannerBasic.html&CTS=' +
Math.ceil(Math.random()*10000);">Click here to create a Basic Store</a>
```

```
<script Language="JavaScript">
   // setting up the window
   var w;
   var launch_str = "Please be patient. The merchant tool will be launched in another
window.";
   var unsupported_browser_text1 = "The merchant tool requires either:";
   var unsupported_browser_item1 = "Netscape Navigator 4.06 or higher";
   var unsupported_browser_item2 = "Internet Explorer 4.01 or higher";
   var unsupported_browser_text2 = "Install the correct browser before creating a
store.";
   var now = new Date();

   // Declaring the basic & advanced URLs as variables
   var basic =
"/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nchs.store_cr
eator.basic.xml&level=Basic&catalogXML=nchs.catalog.simpleCatalog.xml&bannerHTML=regi
sterBannerBasic.html&CTS=" + now.getTime();
   var advanced =
"/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nchs.store_cr
eator.advanced.xml&level=Advanced&bannerHTML=registerBanner.html&CTS=" +
now.getTime();

   function create_basic(){
    create_store(basic)
   }

   function create_advanced(){
    create_store(advanced)
   }

   function create_store(storeurl)
   {
      if ((navigator.appName.indexOf("Netscape")  > -1 &&
parseFloat(navigator.appVersion) >= 4.06) ||
          (navigator.appName.indexOf("Microsoft") > -1 &&
parseFloat(navigator.appVersion) >= 4.0)) {
         w = window.open(storeurl, "MerchantTool",
"resizable=no,scrollbars=yes,status=yes,width=780,height=550,screenX=0,screenY=0,left
=0,top=0");
      } else {

alert(unsupported_browser_text1+"\n"+unsupported_browser_item1+"\n"+unsupported_brows
er_item1+"\n"+unsupported_browser_text2);
      }
   }
</script>
```

Figure 151. Opening store creation wizard in new browser window

10.1.2 Advanced store

The URL to call for the Advanced store creation wizard is:

```
http://<hostname>/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateS
tore&storeXML=nchs.store_creator.advanced.xml&level=Advanced&bannerHTML=re
gisterBanner.html
```

As for the Basic store example, if you open the page in the same browser, you will have to modify the close button on the final creator screen. See Figure 150 on page 226.

The code for opening the advanced store creation wizard in a new window is shown is Figure 151 on page 227 and is similar to the basic store. The call is as follows:

```
<a href="javascript:create_advanced()">Click here to create an Advanced
Store</a>
```

Here is an HTML anchor tag example if you want to call advanced store with the non caching feature.

```
<a href="javascript:window.location =
'/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nc
hs.store_creator.advanced.xml&level=Advanced&bannerHTML=registerBanner.htm
l&CTS=' + Math.ceil(Math.random()*10000);">Click here to create a Advanced
Store</a>
```

HTML showing all these examples for both basic and advanced stores can be found in Appendix H.1, “Provisioning HTML example” on page 379.

10.2 Passing existing customer data to the store creation wizard

Very often, when an existing ISP site has been running for a short time, they already have a number of customers registered and have their own registration page. This page would be common for all the various services that the ISP offers. In the default installation of WCS, the Home Page screen requires a lot of this registration information. If the customer has already typed it in once, it can be irritating to have to type it in a second time. In the example in this section, we will show how to have one common custom registration form for an ISP and make that information available to the store creation wizard. We will also use the direct URL calls to the store creation wizard discussed in the previous sections.

10.2.1 Custom registration page

We created an example registration page that an ISP might have. The customer details can either be typed in or may already exist in a user database held by the ISP. See Figure 152 on page 229.

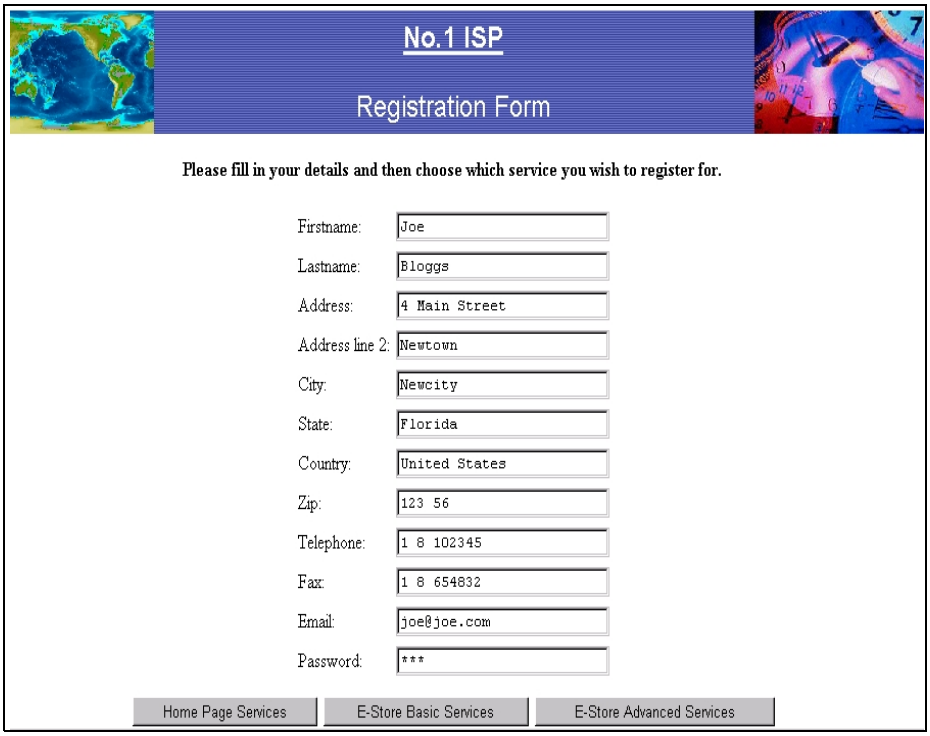


Figure 152. Example ISP registration page

The customers' details are stored in HTML form, whose action is the call to the store creation wizard. The standard parameters required by the wizard are also stored as hidden variables as shown by the following:

```
<form method=post action='/servlet/MerchantAdmin'>
<input type=hidden name=GOTO value="Banner" >
<input type=hidden name=body value="CTnchs.sc.CreateStore" >
<input type=hidden name=storeXML value="" ><
input type=hidden name=level value="" >
<input type=hidden name=catalogXML value="" >
<input type=hidden name=bannerHTML value="" >
<input type=hidden name=CTS value="" >
```

When the customer chooses a Basic e-Store or Advanced e-Store, the appropriate values for these variables are created because each variable has a different value depending on the type of store you want to create. Any spaces or non-alphanumeric characters that exist in the customer data are also converted so that they are sent correctly in the request. This is done with

the convert_params() JavaScript function shown in Figure 153, which describes how we make the call to the basic store wizard.

```
function convert_params(form){
form.firstname.value = escape(form.firstname.value);
form.lastname.value = escape(form.lastname.value);
form.add1.value = escape(form.add1.value);
form.add2.value = escape(form.add2.value);
form.city.value = escape(form.city.value);
form.state.value = escape(form.state.value);
form.country.value = escape(form.country.value);
form.zip.value = escape(form.zip.value);
form.phone.value = escape(form.phone.value);
form.fax.value = escape(form.fax.value);
form.email.value = escape(form.email.value);
form.password.value = escape(form.password.value);
}

function basic_e_store_services(form){

convert_params(form);

form.storeXML.value="nchs.store_creator.basic.xml";
form.level.value="Basic";
form.catalogXML.value="nchs.catalog.simpleCatalog.xml";
form.bannerHTML.value="registerBannerBasic.html";
form.CTS.value= now.getTime();
form.submit();
}
```

Figure 153. Registration page: Basic store wizard functions

This function was called by the following HTML button:

```
<input type=button value="E-Store Basic Services"
onclick="basic_e_store_services(this.form)">
```

Full code for the registration page can be found in Appendix H.2, “Registration example HTML” on page 380.

10.2.2 Displaying e-mail and password automatically on login page

The first store creation wizard page is a login page, and this relates to the file:

/usr/lpp/NetCommerce3/Tools/mpg_template/nchs/store_creator/CreateStore.tem

First, we want to customize this file to show the customer’s e-mail and password typed in on the registration page as the login ID and password. This template file is written in MPG, and the syntax for accessing a parameter is as follows:

$parameters.email$

Where e-mail is the parameter name. To show the e-mail and password parameters in the login screen, we added the code in bold in Figure 154 to the didErrorOccur function, which is run when the page is loaded. Again, WebSphere must be restarted for the changes to take effect.

```
function didErrorOccur()
{
*/
  if ( errormsg != ""  ) {
    /*     alert("$errormsg$");
           document.f1.logonid.value = "$parameters.logonid$";
           document.f1.password.value = "$parameters.password$";
           document.f1.c_password.value = "$parameters.c_password$";
           document.f1.store_name.value = "$parameters.store_name$"; */  endl
  }
/*
// extra code to display email parameter as logonid if it exists
  if ("$parameters.email$" != "UNDEFINED")
    document.f1.logonid.value = "$parameters.email$";
if ("$parameters.password$" != "UNDEFINED"){
    document.f1.password.value = "$parameters.password$";
    document.f1.c_password.value = "$parameters.password$";
    }

  document.f1.logonid.focus();
  return true;
}
```

Figure 154. Displaying e-mail as login ID automatically

The resultant screen is shown in Figure 155.

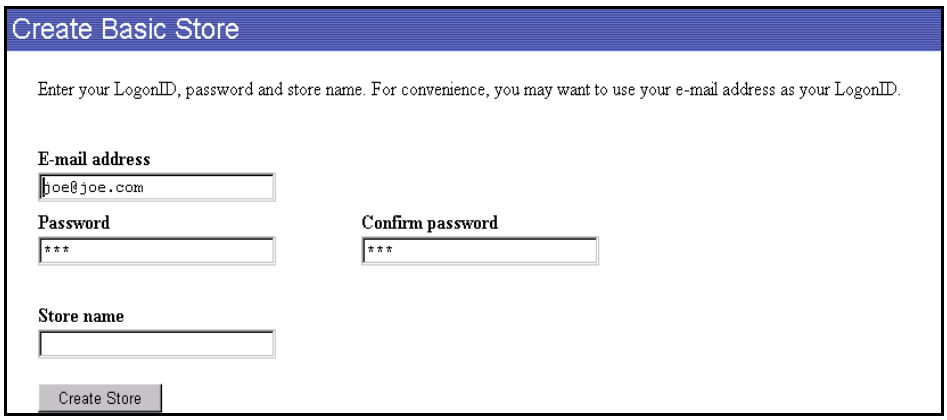


Figure 155. Store creator wizard logon screen

10.2.3 Storing customer details in the store creation wizard

Once the login ID, password, and shop name have been processed, control is passed from the login page to the main store wizard page. Therefore, we also need to pass the customer details parameters again since we want to access them in the wizard's home page screen. To do this, we create the parameters as hidden variables in the CreateStore.tem file.

```
<INPUT TYPE=HIDDEN NAME="firstname" VALUE="$parameters.firstname$" >
<input type=hidden name="lastname" value="$parameters.lastname$" >
<input type=hidden name="add1" value="$parameters.add1$" >
<input type=hidden name="add2" value="$parameters.add2$" >
<input type=hidden name="city" value="$parameters.city$" >
<input type=hidden name="state" value="$parameters.state$" >
<input type=hidden name="country" value="$parameters.country$" >
<input type=hidden name="zip" value="$parameters.zip$" >
<input type=hidden name="phone" value="$parameters.phone$" >
<input type=hidden name="fax" value="$parameters.fax$" >
```

The action of this page's form is the process() section of the same file. It is then necessary to alter the call to the next wizard page by adding our parameters as shown below:

```
-- redirection
/*<SCRIPT>
params = '&firstname='
+escape('$parameters.firstname$')+'&lastname='+escape('$parameters.lastnam
e$');
params +=
'&add1='+escape('$parameters.add1$')+'&add2='+escape('$parameters.add2$');
params +=
'&city='+escape('$parameters.city$')+'&state='+escape('$parameters.state$'
);
params +=
'&country='+escape('$parameters.country$')+'&zip='+escape('$parameters.zip
$');
params +=
'&phone='+escape('$parameters.phone$')+'&fax='+escape('$parameters.fax$');
top.location.href='http://$env.hostname$/servlet/MerchantAdmin?DISPLAY=CTc
ommon.Model&XMLFile=$parameters.storeXML$&email=$parameters.email$' +
params;
</SCRIPT>*/
```

The template file for the next page is found in:

/usr/lpp/NetCommerce3/Tools/mpg_template/common/Model.tem

We can tell this from the DISPLAY parameter set in the call:

DISPLAY=CTcommon.Model

Now, we want to store these parameters in the top page of the wizard so that they will be available for the other panels that will be displayed. We created a JavaScript object called EXT_homepage:

var EXT_homepage = new Object();

Then, we added the customer parameters to this object as shown in bold in Figure 156, and this code was included in the Model.tem file:

/usr/lpp/NetCommerce3/Tools/mpg_template/nchs/store_creator/CreateStore.tem

```
<SCRIPT SRC="/NCTools/javascript/Vector.js"></SCRIPT>

<SCRIPT>
// Code for customer details
var EXT_homepage = new Object();

function setparameter(content){
if (content == "UNDEFINED")
    return "";
else
    return content;
}

EXT_homepage.firstname=setparameter("$parameters.firstname$");
EXT_homepage.lastname=setparameter("$parameters.lastname$");
EXT_homepage.add1=setparameter("$parameters.add1$");
EXT_homepage.add2=setparameter("$parameters.add2$");
EXT_homepage.city=setparameter("$parameters.city$");
EXT_homepage.state=setparameter("$parameters.state$");
EXT_homepage.country=setparameter("$parameters.country$");
EXT_homepage.zip=setparameter("$parameters.zip$");
EXT_homepage.phone=setparameter("$parameters.phone$");
EXT_homepage.fax=setparameter("$parameters.fax$");
EXT_homepage.email=setparameter("$parameters.email$");
//End of custom code

  var model      = new Object();
```

Figure 156. Storing customer details

10.2.4 Displaying customer details in the Home Page screen

Now that we have stored the customer details, they are available to any of the wizard screens. We will show how to access the data using the Home Page screen. The template for the home page for both the advanced and basic stores is the following file:

/usr/lpp/NetCommerce3/Tools/mpg_template/nchs/store_creator/basic/HomePage.tem

We changed the default Initialize() function to display any customer data stored in the object EXT_homepage when loading the page. Also, a new function, setcountry(), was created that set the select box to the country typed in on the registration page. The changed code is shown in bold in Figure 157 on page 235.

```
 //Return the index value of the selectbox option that
// equals customer registration value sentcountry
// otherwise return 0
function setcountry(selectBox, sentcountry){
    found=0;
    for (i=0;i < selectBox.length; i++ ){
        if (selectBox.options[i].text == sentcountry){
            found = i;
            break;}
    }
    return found;
    }

    function initializeState()
    {
      document.f1.firstName.value =
convertFromHTMLToText(stripUndefined(homepage.firstName, top.EXT_homepage.firstname));
      document.f1.lastName.value  =
convertFromHTMLToText(stripUndefined(homepage.lastName, top.EXT_homepage.lastname));
      document.f1.phone.value     =
convertFromHTMLToText(stripUndefined(homepage.phone, top.EXT_homepage.phone));
      document.f1.fax.value       = convertFromHTMLToText(stripUndefined(homepage.fax,
top.EXT_homepage.fax));
      document.f1.email.value     =
convertFromHTMLToText(stripUndefined(homepage.email, top.EXT_homepage.email));

      var found = setcountry(document.f1.merchantCountry,top.EXT_homepage.country );

      var selectedCountry = stripUndefined(homepage.selectedCountry, found);
      document.f1.merchantAddress1.value =
convertFromHTMLToText(stripUndefined(homepage.address1, top.EXT_homepage.add1));
      document.f1.merchantAddress2.value =
convertFromHTMLToText(stripUndefined(homepage.address2, top.EXT_homepage.add2));
      document.f1.merchantCity.value     =
convertFromHTMLToText(stripUndefined(homepage.city, top.EXT_homepage.city));
      document.f1.merchantState.value    =
convertFromHTMLToText(stripUndefined(homepage.state, top.EXT_homepage.state));
      document.f1.merchantZip.value      =
convertFromHTMLToText(stripUndefined(homepage.zip, top.EXT_homepage.zip));
      document.f1.merchantCountry[selectedCountry].selected = true;

      document.f1.aboutus.value =
convertFromHTMLToText(stripUndefined(homepage.aboutus,
"$storeCreatorBasicNLS.HomePageExample1$"));
      document.f1.description.value =
convertFromHTMLToText(stripUndefined(homepage.description,
"$storeCreatorBasicNLS.HomePageExample2$"));
      top.pageAlreadyLoaded = true;
    }
```

Figure 157. Functions to display customer details on the home page screen

So, if any customer details are held in the EXT_homepage object, they will be displayed on the home page screen as shown in Figure 158 on page 236. Remember that in order for any changes made to MPG template files to take effect require WebSphere to be restarted.

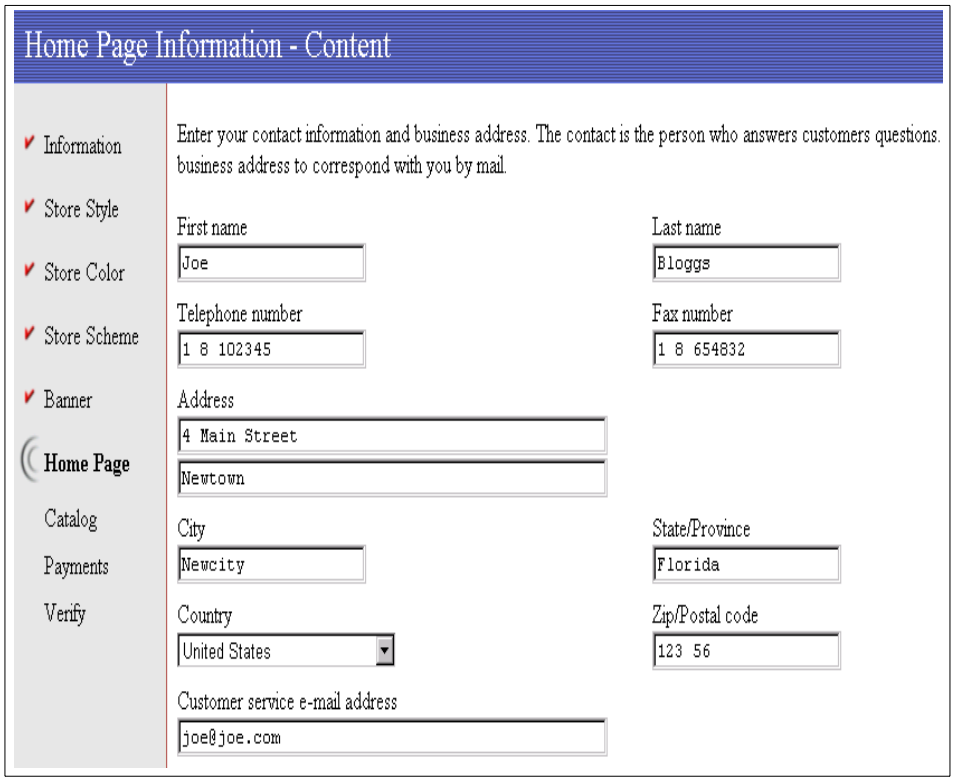


Figure 158. The resultant home page screen

The information in this chapter has described how to include links from Web pages to various WCS pages.

Chapter 11. Multi-language support

Multi-language capabilities are very important, especially in the European market and even in such bilingual countries as Canada, Belgium, and France. In the European Union and Scandinavia, many companies want to use the global nature of the Internet to go beyond national boundaries. This entails a new currency that is supported by WCS SPE as well as a new language for your store. The default language in WCS SPE is the language in which the product was installed.
In this chapter, we want to show how to offer merchants a selection of different languages in which to create their store.

11.1 Background

When a store is created in WCS SPE a number of HTML files are created containing hardcoded textual information. These are found in the following files:

```
/usr/lpp/internet/server_root/pub/<merchant.mestname>/homepage.html
/usr/lpp/internet/server_root/pub/<merchant.mestname>/navigationBar.html
```

The default language for this is the installed language version of WCS. The content in homepage.html can be changed through the store creation wizard. But not the navigationBar.html. Its content can be changed through the merchant wizard, but it would be more professional if the navigation bar was created in the new language chosen. To do this, we must send an extra XML parameter to the final creation process in the store wizard as in the variable, enXML, shown in Appendix I.1, “displaylangs.html” on page 383.

We also need to store language choice, for which we have chosen custom field merchant.mefield1.

For the Net.Data macro files, the translation text is stored in the following file:

```
/usr/lpp/NetCommerce/macro/en_US/<StoreModel>/translation_text.inc
```

11.2 Store creation wizard multi-language panel

We created a new panel for the store wizard, shown in Figure 159 on page 238. Implementation details are listed in Appendix I, “Multi-language samples” on page 383. Here, the merchant can choose a language. This then stores the appropriate XML parameter in the top frame and also calls the new command, `langAdd`, which stores this language value in merchant.mefield1.

Figure 159. Language selection panel

To implement the new navigationBar.html, we made the following changes to the file:

```
/usr/lpp/NetCommerce3/Tools/mpg_templates/nchs/store_creator/advanced/Verify.tem
```

We also added a new XML parameter:

```
<FORM NAME="f1" METHOD=POST
ACTION="http://$env.hostname$/servlet/tempServlet">
<INPUT TYPE=HIDDEN NAME="PROCESS" VALUE="CTnchs.sc.advanced.Verify">
<INPUT TYPE=HIDDEN NAME="XML" VALUE="">
<INPUT TYPE=HIDDEN NAME="XML" VALUE="">
<INPUT TYPE=HIDDEN NAME="XML" VALUE="">
<INPUT TYPE=HIDDEN NAME="XML" VALUE="">
<INPUT TYPE=HIDDEN NAME="XML" VALUE="">
<INPUT TYPE=HIDDEN NAME="XML" VALUE="">
```

Set it to the value in the top frame in the validatePanelSubmit() JavaScript function:

```
document.f1.XML[5].value = top.langElements.XML;
```

And, lastly, in the Process() section of the MPG, we added it as a parameter to the settings file:

```
Model navigationBar
x = settings.settings.put("navigationBar", navigationBar)
```

These parameters are then sent to the final screen when the store is created, and a new language navigation bar is created as shown in Figure 160 on page 239.

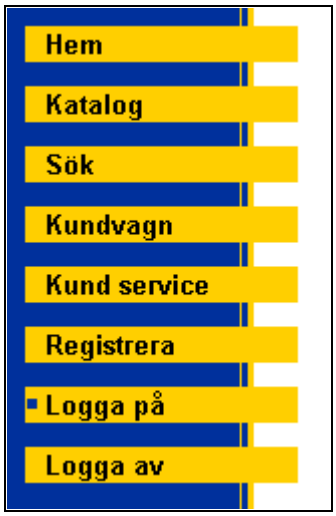


Figure 160. New language navigation bar

11.3 Multi-language macros

We want to replace the default translation file with our new custom files. We create these file as follows:

```
/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/sv_SE/translation_tex
t.inc
/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/en_US/translation_tex
t.inc
/usr/lpp/NetCommerce3/macro/common/ClassicStoreModel/de_DK/translation_tex
t.inc
```

Sample content of this file is shown below for the logon screen:

```
@DTW_ASSIGN(BUT_LOGOFF, "Logga av")
@DTW_ASSIGN(BUT_LOGON               , "Logga på")
@DTW_ASSIGN(LBL_LOGONID             , "Användar ID:")
@DTW_ASSIGN(LBL_LOGONID_OLD         , "Användar ID:")
@DTW_ASSIGN(LBL_PASSWORD            , "Lösenord:")
@DTW_ASSIGN(TXT_TITLE_LOGON           , "Logga på")
@DTW_ASSIGN(TXT_INSTR_LOGONFORM  , "")
```

Shown in bold in Figure 161 on page 240 are the changes we made to the default logon macro to implement the new language. In this code, we select the language from the merchant's database and then use that to include the appropriate language file. The default logon macro is:

```
%function(dtw_odbc) GET_MERCHANT_LANG() {
    SELECT mefield1 as lang
    FROM merchant
    WHERE merfnbr = $(MerchantRefNum)

    %REPORT{
        @DTW_ASSIGN(lang,V_lang)

    %}

    %MESSAGE{
        100: { @DTW_ASSIGN(lang,"en_US") %} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B
> GET_ADDRESS_REF()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %}
    %}
%}


%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT {
<HTML>

<HEAD>
    <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1999 01:01:01 GMT">
    %INCLUDE "ClassicStoreModel/browser_addrsettings.inc"
</HEAD>

<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)" TEXT="$(TextCol)"
LINK="$(LinkCol)" VLINK="$(VLinkCol)" ALINK="$(ALinkCol)">

@GET_MERCHANT_LANG()
%include "/ClassicStoreModel/$(lang)/text.inc"
```

Figure 161. Enabling new language schema in common macros

The new macro will appear as shown in Figure 162 on page 241.

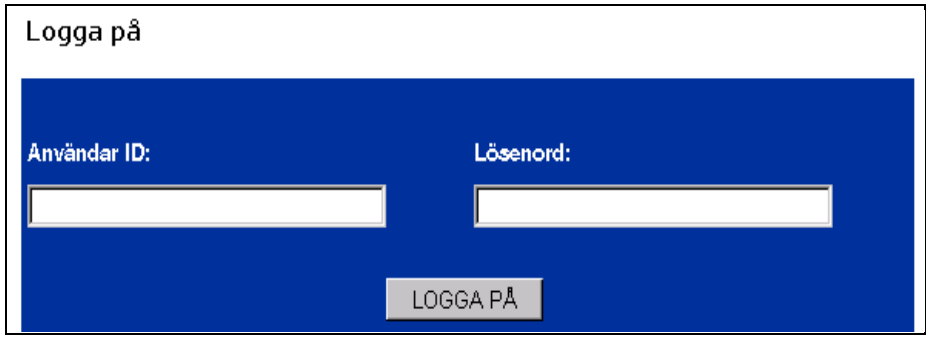


Figure 162. Language-specific logon

Appendix A. Multi purpose code generation language

The purpose of this document is to describe a template-driven framework that may be used to facilitate the generation of code. This framework has many applications, from simple form letters or HTML pages to complex programs or class libraries created by application builders.

A.1 Introduction to MPG

MultiPurpose Generator (MPG) is a utility that may be used to generate output. The output may be anything from form letters to HTML pages to complex C++/Java code. MPG splits the generation into two components: The model and the template. The model is the set of external data (variables) that is used to generate the output. The template is the logic that describes how the output is to be generated.

Consider a letter of confirmation that an employer sends to confirm that they have received an application for employment. Rather than personally writing a letter to every person that applied, the company would probably issue a standard letter and simply change the name, address, position applied for, and so on. In this case, the standard letter would be the template, and the model would consist of the following static information:

Model:

name.first:	Sally
name.last:	Smith
address.line1:	2400 Bayview Ave, Apt 23
address.line2:	North York, Ont
address.postal_code:	M4N 1JS
position:	Development Analyst
internal_contact:	Dave Johnston
contact_position:	Human Resources Directory
date:	June 21, 1997
cc_list[0]:	Fred Smith
cc_list[1]:	Barney Noble

Template:

```
form_letter()
```

```
{
 Model name, address, position, internal_contact, contact_persion, date,
cc_list

 /*
Enterprise Software

$date$

$name.first$ $name.last$
$address.line1$
$address.line2$
$address.postal_code$

Dear $name.first$:

On behalf of the company I would like to thank you for applying for the
position of $position$. Blah, Blah....

Sincerely,


$internal_contact$
$contact_position$

cc: */
 repeat(cc_list | middle (", ") ) {
  cc_list
 }

}
```

The above example reveals some of the elementary syntax of the template language. The body of the letter is enclosed with the delimiters, /* and */, as free-form text. For convenience, the writer of the template has escaped from the free-form text using a $ sign to substitute a variable from the external model (This is the same as /* Enterprise Software */ date /* .... */). Variables are pulled in from the model by declaring them as Model var1, var2, and so on.

A.1.1 Why MPG?

This is a question that has been asked often, and it is best answered by starting at the root from which this language evolved.

MPG evolved from my work on a previous project: Data Access Builder for C++/Java (DAX). DAX was an application that was used to map database tables and SQL result sets to object-oriented classes. Once the user had defined a mapping, the builder would generate classes (in the form of C++ or Java code) that could be used to access the database. The logic for generating these classes was originally C++ code. During development, the developer would make changes to the code, rebuild the application, run it, and examine the output. This became a very tedious process, especially when making small changes, such as updating a comment. The syntax of the C++ language made the code generation rather awkward. Static text could not span more than one line without ending the string with an " (and don't forget to put \n!). Substituting variables often required a string conversion and syntax, such as ..." + new IString(variable) + "... The looping structures (while, for) were not well-suited for generating function calls (',' must be between variables, need to keep track of an external iterator). In general, handling lists proved awkward. Over time (with many developers editing the same code), the model/view separation in the code started to deteriorate.

It was any of these many other issues that started the evolution of MPG. First, the static text was moved into an external file to make for easy "simple" modifications. The text file eventually served as an outline for the generated code. It became natural to add certain "primitive" constructs to this file, such as conditional and looping structures. This text file eventually evolved into the MPG template.

The following summarizes some of the key features of MPG that make it a better alternative to using traditional languages, such as C++/Java:

1. Clear separation between model and view
2. Simple language designed for code generation
    1. Easier syntax
        1. Free-form text
        2. No statement terminators
        3. Output stream is implicit
    2. Weak typing
        1. Datatype conversions are automatic
        2. No type casting
    3. Specially designed constructs
        1. Repeat loop
        2. List manipulation

3. Designed for the common ‘code generation’ case

4. Special additions
    1. Counters
    2. SQL Queries
    3. SQL Statements
5. String manipulation
    1. Built-in transformations for string formatting
    2. Built-in support for currency formatting

3. Interpreted templates (no need to compile)
    1. Quicker development life cycle
4. Security control
    1. Direct control over the model that is accessible from the template
    2. No file manipulation
    3. No network access
    4. Database access can be turned on/off
    5. Scope is limited to the given print stream and the external model. (No other I/O can occur)

A.2 Data model

In this section, we will describe how to create the model that is to be used in the template.

A.2.1 Declaring model variables

In order to use variables from the model, they must be declared in the template. A declaration is prefaced with the keyword Model. Multiple entries may be used from the model by separating them with commas. Model declarations may occur inside procedures or at the global scope (accessible in all procedures).

Example:

```
main()
{
Model env, merchant
...
}
```

In the above template, the programmer has declared two variables that will be extracted from the model: *env* and *merchant*. At runtime, the generator will extract the variables that have been mapped to the keys (*env, merchant*) from the model (Hashtable) and map them to *env* and *merchant* in the template.

A.2.2 Creating the model

Since MPG has been written in Java, its data model consists of Java objects. The model that is passed to the template is a Hashtable that contains a mapping of the name of the variable to the Java object that it represents. Some of the most common types of Java objects used in the model are String, Boolean, Double, Vector, and Hashtable.

Example:

```
Hashtable model = new Hashtable();
```

A.2.2.1 Using regular Java objects

Regular Java objects (that is, from package java.lang) are typically used as single values in the model. Objects of type *java.lang.String* are the most commonly used. Numbers (*java.lang.Integer*, *java.lang.Double*), *java.lang.Boolean*, and so on are also quite common. In general, MPG uses the *toString()* method on these objects when generating the output.

Example:

```
model.put("name", "Jane Smith");
model.put("today", new Date());
model.put("flag", Boolean.TRUE);
model.put("number_of_items", new Integer(20));
```

A.2.2.2 Using hashtables

Objects of type *java.util.Hashtable* may be used in MPG to create structures. For example, the address variable consists of a Hashtable that contains other variables (in this case, variables of type *String*). The variables in the Hashtable may be directly accessed by their keys. The Hashtable can, of course, contain other Hashtables; this allows infinite levels of structures and substructures.

For example, the model used in the form letter template (see introduction) may have been created as follows:

```
// name substitutions
Hashtable name = new Hashtable();
name.put("first", "Sally");
name.put("last", "Smith");
model.put("name", name);
```

```
// address substitutions
Hashtable address = new Hashtable();
address.put("line1", "2400 Bayview Ave, Apt 23");
address.put("line2", "North York, Ont");
address.put("postal_code", "M4N 1JS");
model.put("address", address);
```

A.2.2.3 Using vectors and arrays

The *java.util.Vector* class and arrays of type *Object[]* are also treated specially in MPG. The template allows you to loop through elements of these variables. Consider a situation where we are generating the private members of a Java class. The data will consist of two lists, a list containing the member names (member_names) and a list containing the types of the members (data_types):

Model:

```
// member names
Vector member_names = new Vector();
member_names.addElement("var1");
member_names.addElement("var2");
member_names.addElement("var3");
model.add("member_names", member_names);

// data types
Vector data_types = new Vector();
data_types.addElement("int");
data_types.addElement("String");
data_types.addElement("double");
model.add("data_types", data_types);
```

Template:

```
class_definition()
{
   Model member_names, data_types
   /*
   public class A
   {
       public A() {} */ endl
       repeat( member_names, data_types ) {
           /* private $data_types$ $member_names$; */ endl
       }
       /*
   }*/
}
```

Output:

```
public class A
{
   public A() {}
   private int var1
   private String var2
   private double var3
}
```

The preceding example illustrates the use of single value variables as elements in the Vector. Vectors and arrays may contain any type of object including other Vectors and Hashtables.

A.2.2.4 Using Java Beans

Java Beans may also be used in the model. The bean properties may be accessed directly as properties of the variables to which they are mapped. For example, consider a Java class, Merchant, that has a property, called refno. The Merchant class would have a method, getRefno(), that would return the value of the property. If an object of type Merchant were mapped to the variable name, merchant, in the model, it may be referenced in the template as merchant.refno.

At runtime, the template would end up calling Merchant.getRefno() to resolve the value of merchant.refno.

Parameterless methods may also be invoked on the beans. For example, if the Merchant class has a method update(), it may be invoked in the template as merchant.update().

A.2.3 Creating the model from a file

The model may be declared in a file and created using the model parser. The model parser reads in the file and builds the model that may be used to pass to a template. The following is the grammar that is used to define the model:

declarations ::= declaration [declaration] ...

declaration ::= var_name = value

value :== single_value | table_value | list_value

single_value :== string | number | boolean

table_value :== { declaration [, declaration] ... }

list_value :== { value [, value] ... }

var_name :== letter [letter | digit | _] ...

string :== “characters“ | ‘characters‘ *Note: use backslash to obtain special characters (ie \n)*

number :== digits | digits . digits

boolean :== true | false

letter :== a-z A-Z

digits :== digit [digit]...

digit :== 0-9

character :== *any valid character*

Example:

```
--
-- The following describes the model used in the introductory sample
--
name={ first='Sally', last="Smith" }
address={ line1 = "2400 Bayview Ave, Apt 23",
         line2="North York, Ont",
     postal_code="M4N 1JS" }
position="Development Analyst"
internal_contact= "Dave Johnston"
contact_position="Human Resources Directory"
date="June 21, 1997"
cc_list = [ "Fred Leary", "Barney Noble" ]
```

A.3 Language elements

The preceding sections informally introduced some of the elementary syntax of the language. In this section, we will explore, in detail, the syntax and directives of the language. Syntactically, the MPG language resembles C, C++, and Java. This resemblance was intentional to make it easier for experienced programmers to quickly get up and running using MPG.

A.3.1 Lexical structure

The lexical structure of a programming language is the set of elementary rules that specify how you write programs in the language. This low-level

syntax specifies details, such as what variable names look like, comments, and how statements are separated.

A.3.1.1 Case-sensitivity

MPG is a case-sensitive language. Keywords, variable names, and procedure names must be typed with consistent capitalization. For example, the variable, x, is entirely different from the variable, X.

A.3.1.2 Whitespace

The MPG parser ignores spaces, tabs, and new lines that appear between tokens in the templates. Note that this does not include white spaces that are included in strings (delimited by ‘ and “) and whitespace that exists inside the free-form delimiters (/* and */).

A.3.1.3 Comments

There are two types of comments in MPG. The first type of comment spans a single line. This comment starts with the comment delimiter --. Anything appearing after -- on the same line will be treated as a comment and ignored by the parser. The second type of comment spans multiple lines and is often useful for removing pieces of code for testing purposes. These types of comments start with /- and end with -/.

Note that comments are not allowed within literal strings and within the free-form delimiters.

A.3.1.4 Statement terminators

Many languages, such as C/C++ or Java, use a semi-colon (;) to act as a statement terminator. In MPG, there are no statement terminators. Semicolons may be added to statements but will be ignored.

A.3.1.5 Literals

A literal is a data value that appears directly in the template. Literals consist of numbers, strings, and Boolean values (true, false). Numbers may be integers or floating point values. Floating point values use a period as the decimal separator. Strings are delimited between ‘’ or “” and must occur on a single line. All characters between the delimiters are part of the string. Special characters may be added to the string by use of the backslash (\). For example, to add a new line to a string literal, use \n (‘line1\nline2’). The Boolean values, *true* and *false*, may also appear as literals in the template.

A.3.1.6 Free-form text

One element in particular that distinguishes the syntax of MPG from other programming languages is the ability to add freeform text to a template.

Free-form text may include any text (including newlines). Free-form text is delimited by C-style comments /* and */. These delimiters were purposefully chosen to allow the text to appear in a different color if the programmer is using a color-coded C-style editor, which is very common nowadays.

For convenience, statements may be embedded in freeform text by delimiting them in dollar signs ($). This is often used to substitute a variable inside the text. The tilde character (~) may be used to escape characters, such as the dollar sign, in cases where it is not meant to escape the free-form text.

A.3.1.7 Identifiers

An identifier is simply a name. In MPG, identifiers are used to name variables and functions. The rules for legal identifiers are similar to those of C. The first character must be a letter and may be followed by any combination of letters, numbers, and underscores (_). Note that identifiers can only contain ASCII characters. Unicode or Latin-1 characters are not allowed.

A.3.1.8 Reserved words

There are a number of reserved words in MPG. These are words that you cannot use as identifiers. The following is a list of reserved words in MPG:

if	Var	repeat
else	Query	while
contains	Statement	before
startsWith	Counter	after
endsWith	List	middle
endl	Stack	condition
true	return	stop
false	continue	include
Model	break	

A.3.2 Declarations

Before any variables may be used in the template, they must be declared. There are four different types of variables: Model variables, SQL queries, SQL statements, and regular variables. Declarations may occur anywhere in the template and are valid in the scope that they are declared and any subscopes. Variables are declared by starting with a keyword that identifies the type of variable. These keywords are:

- **Model**
- Var

- Query
- Statement

The keyword is followed by the name of the variable. Multiple variables may be declared at the same time by separating them with commas (,).

Examples:

```
Model env, merchant
Var x
Query query1, query2
```

A.3.2.1 Variable scope

The scope of a variable defines the section of code where the variable exists. A scope is defined by a starting { and an ending }. Each procedure has a different variable scope. Even repeat loops and *if* statements can have variable scopes. Any variables that are defined outside of a procedure are considered “global”. This means that they are accessible to all parts of the template.

A.3.3 Data types

The MPG template language itself only supports six data types: Numbers, strings, and Boolean values (true, false), counters, stacks, and lists. These are the types of data that may appear as literals in the template. The external model, however, supports all Java objects. The objects in the external model may be manipulated through there defined methods and properties or converted to strings (typically using the *toString()* method).

A.3.4 Expressions and operators

Expressions and operators are very similar to what you will find in C/Java.

An expression is a “phrase” that the MPG runtime can evaluate to produce a value. The simplest expressions are literals or variable names. The value of a literal is the literal itself. The value of a variable expression is the value to which the variable refers. More complex expressions may be constructed using operators.

Operators are operations that apply to one or more expressions. MPG has two types of operators: Unary operators, which apply to a single expression, and binary operators, which apply to two expressions. The following table describes all of the operators in MPG. Included in this table is the precedence of the operators. The precedence determines the order in which a set of

operations will occur. In an expression where the precedence is the same for all operators, it is evaluated from left to right

Table 19. MPG operators

P	Operator	Type	Operand type(s)	Action performed
7	.	binary	variable, property	Accesses a property or method on the variable
7	[ ]	binary	list, integer	Accesses an element in a list
7	[ ]	binary	variable, expression	Accesses property of variable (similar to an associative array)
8	( )	unary	expression	Makes the expression the highest precedence
7	++	unary	variable (number)	Increments the number by 1
7	++	unary	variable (list)	Increments the list's internal counter
7	--	unary	variable (number)	Decrements the number by 1
7	--	unary	variable (list)	Decrements the list's internal counter
5	+, -	binary	numbers	Addition, subtraction
6	*, /	binary	numbers	Multiplication, division
5	+	binary	strings	Concatenation
2	&&	binary	booleans	Logical AND
2	||	binary	booleans	Logical OR
4	!	unary	boolean	Logical complement (NOT)
3	==	binary	any	Test for equality
3	!=	binary	any	Test for inequality
3	>, >=	binary	numbers or strings	Greater than, greater than or equal

P	Operator	Type	Operand type(s)	Action performed
3	<, <=	binary	numbers or strings	Less than, less than or equal
3	contains	binary	strings	True if left arg contains right arg
3	startsWith	binary	strings	True if left arg starts with right arg
3	endsWith	binary	strings	True if left arg ends with right arg
1	=	binary	variable, expression	Assigns value of expression to variable
1	+=	binary	variable, expression	Appends the value of the expression to the variables current value (for numbers the numeric value is added)
7	|	binary	expression, transformation	Applies the given transformation to the expression
8	new	unary	type	Creates a new object (currently: Stack, Counter, List)

A.3.5 Statements

As described in the preceding section, expressions are phrases that may be evaluated to produce a value. The general effect of an expression is to output the value to the print stream and/or produce side effects, such as moving the internal cursor on a list. To add logical flow to a template, you need to use statements.

A.3.5.1 The *if* statement

The if statement is used to execute statements or expressions conditionally. This works the way you would expect if you are a C/Java programmer:

Form 1:

if (expression) statement

or optionally:

```
if (expression) {
  statements
}
```

Form 2:

```
if (expression) statement
else statement
or optionally:
if (expression) {
 statements
}
else {
 statements
}
```

If the evaluated expression is *true* or evaluates to the string value *"true"*, the statement (or block of statements) associated with the *if* is executed. If an *else* condition is given and the expression does not evaluate to *true* (or string value *"true"*) the statement (or block of statements) associated with the *else* is executed.

Multiple conditional situations can be handled by chaining *if* statements off of the *else* clauses.

Example:

```
if (env.isNetscape_v2) {

}
else if (env.isNetscape_v3) {

}
else if (env.isNetscape_v4) {

}
...
```

A.3.5.2 The *repeat* statement

Looping is achieved through the use of the *repeat* statement. Options to the repeat statement consist of two parts.

In the first part, known as the repeat list, the items to iterate over are given. The items are variables that are usually lists, counters, or SQL queries. The variables in this list must be separated by commas (,). All repeatable elements in MPG maintain an internal iterator. The first time through the repeat loop, the internal iterator of each element is reset. This is equivalent to calling the *reset()* method on each variable. Through each iteration of the loop, each element increments its internal iterator using the *increment()* method, which is equivalent to the ++ operator. The loop ends when one of the members in the repeat list reaches its last value of the stop condition (see the following) is reached.

The second part of the repeat statement consists of blocks of code to execute before and after the loop, code to execute in-between each iteration, conditions to meet before executing the block of the repeat statement, and a stop condition. These options are separated from the repeat list by using a vertical bar (|).

The *before* option consists of a block of code preceded by the keyword, *before*. The block of code is executed before the first iteration of the loop and is only performed if the body of the loop is executed at least once. Similarly, the *after* option is performed upon completion of the loop only if the body was executed at least once. The *middle* directive may be used to perform operations between each iteration. A common use of this option is to generate an argument list that is separated by commas:

```
repeat( method.args | before { /*$method.name$( */ }
                      middle /*, */
                      after /* ); */ )
{
 args.name
}
```

The *condition* option may be used in order to avoid certain combinations of the repeat list occurring. The body of the loop will only be executed when the expression in the condition evaluates to “true”. If, in the above example, we wanted to generate code that invokes a method only with the arguments that are primitive types:

```
repeat( method.args | before { /*$method.name$( */ }
```

```
                middle /*, */
                after /* ); */
                condition( isPrimitive(args.type) ) )
{
 args.name
}
```

Note that the middle block is only executed between the arguments that meet the condition. If no arguments are primitive, the before and after blocks are not executed.

The *stop* option halts the execution of the loop when its condition evaluates to true. For example, if it was necessary to generate the method call with the first three arguments, the following code could be used:

```
Var i = new Counter(1,1)

repeat( method.args, i | before { /*$method.name$( */ }
                middle /*, */
                after /* ); */
                stop( i == 4) )
{
 args.name
}
```

A.3.5.3 The *while* loop

While loops may be used to repeat a segment of code while a certain condition remains true. This is used the in the same way as the while loop in C.

Example:

```
while( i > 0 ) {
  i = i-1
}
```

A.3.5.4 The *return* statement

The *return* statement is useful for terminating the execution of a procedure. As soon as a procedure encounters a return statement, the procedure ends and returns to its caller.

A.3.5.5 The *break* statement

The *break* statement is useful for terminating the execution of a loop. As soon as a loop encounters a break statement, the loop terminates.

A.3.5.6 The *continue* statement

The *continue* statement is useful for terminating the current iteration of a loop. As soon as a loop encounters a continue statement, the loop starts its next iteration.

A.3.6 Procedures

A procedure is a piece of MPG code that is defined once in a template and can be executed many times. Procedures may be passed arguments specifying the value(s) upon which the procedure is to operate. Procedures are defined as follows:

```
procedure_name(arg1, arg2)
{
 <procedure body>
}
```

Procedures are invoked by following the procedure's name with an optional comma-separated list of arguments within parentheses. The following are examples of procedure invocations:

```
display_category()
x = generate_select()
redirect("http://www.mystore.com/" + subdir + "/index.html")
```

The second example illustrates how to intercept the output of a procedure. By default, the output of a procedure (that is, the code that it generates) is sent to the output stream. By assigning the procedure call to a variable (*x = generate_select()*) the output is intercepted (in this case, placed in variable x).

A.3.7 Transformations

Transformations are a set of utility functions that are used for formatting (transforming) the output. A transformation may be applied to an expression by placing a vertical bar (|) after it, the name of the transformation and the argument list. The argument list is dependant on the transformation.

Any number of transformations may be applied to an expression by chaining them together and separating them with vertical bars (|). They will be evaluated in left-to-right order, and the result of each transformation is passed to the next transformation in the list.

Example:

member.name|lowerCase()|change("get", "set")|upperCase(4,1)

Table 20 describes all of the transformations.

Table 20. Transformations

Transformation	Description
upperCase()	Converts expression to upper case
upperCase(pos)	Converts expression to upper case starting at the given position
upperCase(pos, length)	Converts expression to upper case starting at the given position for the given length
lowerCase()	Converts expression to lower case
lowerCase(pos)	Converts expression to lower case starting at the given position
lowerCase(pos, length)	Converts expression to lower case starting at the given position for the given length
substring(pos)	Returns the portion of the expression start at the given position
substring(pos, length)	Returns the portion of the expression start at the given position for the given length
upperCase()	Converts expression to upper case
upperCase(pos)	Converts expression to upper case starting at the given position

Transformation	Description
upperCase(pos, length)	Converts expression to upper case starting at the given position for the given length
isUpperCase()	Tests to see if the expression is all upper case
isUpperCase(pos)	Tests to see if the expression is all upper case starting at the given position
isUpperCase(pos, length)	Tests to see if the expression is all upper case starting at the given position for the given length
isLowerCase()	Tests to see if the expression is all lower case
isLowerCase(pos)	Tests to see if the expression is all lower case starting at the given position
isLowerCase(pos, length)	Tests to see if the expression is all lower case starting at the given position for the given length
isDigit()	Tests to see if the expression is all digits
isDigit(pos)	Tests to see if the expression is all digits starting at the given position
isDigit(pos, length)	Tests to see if the expression is all digits starting at the given position for the given length
remove(pos)	Removes the substring starting at the given position
remove(pos, length)	Removes the substring starting at the given position for the given length
insert(str)	Inserts the given string at the beginning of the expression
insert(str, pos)	Inserts the given string at the given position
change(pattern, str)	Replaces all occurrences of the *pattern* with the replacement string *str*
change(pattern, str, max)	Replaces up to *max* occurrences of the *pattern* with the replacement string *str.*

Transformation	Description
changeToken(pattern, str)	Tokenizes the string and replaces all tokens matching pattern with the replacement token str. **(TBD)**
changeToken(pattern, str, max)	Tokenizes the string and replaces up to *max* all tokens matching pattern with the replacement token str.
length()	Returns the length of the string as an integer.
strip()	Strips leading and trailing whitespace
strip(chars)	Strips leading and trailing characters in the set *chars*
stripLeading()	Strips leading whitespace
stripLeading(chars)	Strips leading characters in the set *chars*
stripTrailing()	Strips trailing whitespace
stripTrailing(chars)	Strips trailing characters in the set *chars*
occurencesOf(pattern)	Returns the number of occurrences of the specified pattern
indexOf(pattern)	Returns the index of the given pattern (index starts at 0)
isDigit()	Tests to see if the expression is all digits
isDigit(pos)	Tests to see if the expression is all digits starting at the given position
isDigit(pos, length)	Tests to see if the expression is all digits starting at the given position for the given length
remove(pos)	Removes the substring starting at the given position
reverse(string)	Reverses a string
round()	Rounds the number to an integer
round(precision)	Rounds the number to the given number of decimal places.
floor()	Returns the floor of the number (0 decimal places)

Transformation	Description
floor(precision)	Returns the floor of the number to the given number of decimal places.
ceil()	Returns the ceiling of the number (0 decimal places)
ceil(precision)	Returns the ceiling of the number to the given number of decimal places.
abs(number)	Returns the absolute value of the given number
format(locale)	Formats a number in the given locale
format(precision)	Formats a number to have the given precision (appends 0s if necessary)
format(precision, locale)	Formats a number to have the given precision (appends 0s if necessary) in the given locale
toNumber()	Converts the expression to a number (using the default locale)
toNumber(locale)	Converts the expression to a number using the given locale
toDouble(locale)	Converts the expression to a double using the given locale
toInteger(locale)	Converts the expression to a integer using the given locale
toBoolean()	Converts the expression to a Boolean
toString()	Converts the expression to a string
toTimestamp(value)	Converts the value to a timestamp. If value is a number, it contructs a Timestamp object with the long value; otherwise, it assumes the string value is in the format: YYYY-MM-DD-HH.MM.SS.SSSSSS
toTime(value)	Converts the value to a Time. If value is a number, it contructs a Time object with the long value; otherwise, it assumes the string value is in the format: HH.MM.SS

Transformation	Description
toDate(value)	Converts the value to a java.sql.Date. If value is a number, it contructs a Date object with the long value; otherwise, it assumes the string value is in the format: YYYY-MM-DD
toJavaScript(value)	Converts the value so it may be assigned to a Java script variable (replaces ' with \', \n with \\n, " with \\")
format_date()	Formats a date using the default locale
format_date(locale)	Formats a date using the given locale
format_time()	Formats a time using the default locale
format_time(locale)	Formats a time using the given locale
format_timestamp()	Formats a timestamp using the default locale
format_timestamp(locale)	Formats a timestamp using the given locale
currency(locale)	Formats the number as a currency for the given locale
currency_HTML(locale)	Same as currency, formats for HTML display
currencySET(symbol)	Formats the number for the given SET symbol
currencySET_HTML(symbol)	Same as currencySET, formats for HTML display
currencyNoSymbols(locale)	Same as currency, removes currency symbols.
currencyNoSymbolsSET(setCode)	Same as currencySET, removes currency symbols
currencyNoSymbolsSET_HTML(setCode)	Same as currencyNoSymbolsSET, formats for HTML display
toJavaScript(value)	Converts the value so it may be assigned to a JavaScript variable (replaces ' with \', \n with \\n, " with \\")
format_date()	Formats a date using the default locale

Transformation	Description
format_date(locale)	Formats a date using the given locale
format_time()	Formats a time using the default locale
format_time(locale)	Formats a time using the given locale
format_timestamp()	Formats a timestamp using the default locale

A.3.8 4.10 including other templates

Code from other templates may be reused by including them in the current template. This allows you to reference other procedures or global variables defined outside of the template. The following is an example of how you would include another template:

include "common/address_formats.tem"

The **include** directive finds the given template by searching the template path. This is a file system path (or paths) that is searched to find the given template. Subdirectories may be searched by specifying the subdirectory in the include directory (as shown). The directory separator is /. This is the same across all platforms (including Windows platforms).

Appendix B. ShopperGroup code samples

This appendix contains source code listing for programs used in Chapter 5, “Shopper groups” on page 61

B.1 RegWrapper source code

```
/****************************************************************************
** Licensed Materials - Property of IBM                                    **
**                                                                         **
** 5697-D24                                                                **
**                                                                         **
** (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.             **
**                                                                         **
** US Government Users Restricted Rights - Use, duplication or disclosure  **
** restricted by GSA ADP Schedule Contract with IBM Corp.                  **
****************************************************************************/

/*
#############################################################################

regWrapper Command for redbook

Date:2000/04/04

Purpose : Wrapper for RegisterNew cmd that relates new shopper to merchant

Usage: /cgi-bin/ncommerce3/regWrapper?merchant_rn=<>&regtype=<>
    &url=<>&... normal params for RegisterNew

History
=======
  Version  Date      Who  Comment
  -------  --------  ---  -------

#############################################################################
*/

#include "objects/objects.pch"

//=============================================

#ifdef __TRACE_COMMANDS__
  typedef TraceYes Trace;
#else
  typedef TraceNo  Trace;
#endif
static Trace trace("MY_COMMANDS ("__FILE__")");
//=============================================


#if defined(WIN32)
 #define __EXPORT_MODE__ __declspec(dllexport)
#elif defined (AIX)
 #define __EXPORT_MODE__
#endif


//=========================================================================================
```

```
   // variables used by the Process function
   String merchantRefNum;
   String _VAL_url;

   // optional parameter
   String _VAL_regtype;

   //NVP_names declarations
   static const StringWithOwnership NVP_merchant_rn("merchant_rn");
   static const StringWithOwnership NVP_url("url");

   // optional
   static const StringWithOwnership NVP_regtype("regtype");


////////////////////////////////////////////////////////////////////////////////
///////////
//
// regWrapper
//
////////////////////////////////////////////////////////////////////////////////
///////////
class __EXPORT_MODE__ regWrapper : public NC_Command
{
   static const ClassName _STR_ThisClass;


public:
   regWrapper(void)
   {
      // Trace the function call
      Trace::Tracer T(_STR_CONSTRUCTOR, _STR_ThisClass);

   }


   virtual bool Initialize(void)
   {
      // Trace the function call
      Trace::Tracer T(_STR_Initialize, _STR_ThisClass);

      return true;
   }


   virtual ~regWrapper(void)
   {
      // Trace the function call
      Trace::Tracer T(_STR_DESTRUCTOR, _STR_ThisClass);

   }

   void  operator delete( void* p ) { ::delete p; }
   virtual NC_Command* Clone(void) { return new regWrapper; }

public:
   virtual void FailedRegistration(NC_RegistrationID& RegID, const ErrorMsg_Reg* Err)
   {
      // Trace the function call
      Trace::Tracer T(_STR_FailedRegistration, _STR_ThisClass);

      if (Err == &_ERR_REG_UNEXPECTED_OBJ)
      {
```

```
         error << indent << "ERROR: regWrapper not accepted by the manager" << endl;
      }
      else if (Err == &_ERR_REG_DUPE_ID)
      {
         error << indent << "ERROR: regWrapper: another command with the same name has
already been registered" << endl;
      }
      else if (Err == &_ERR_REG_OBJ_NOT_FROM_DLL)
      {
        error << indent << "ERROR: regWrapper is packaged in a different DLL then the one
registered in the DB" << endl;
      }
      else
      {
         error << indent << "ERROR: regWrapper unknown registration error" << endl;
      }

      error << indent << "ERROR: regWrapper's registration failed" << endl;
   }


   virtual bool CheckParameters(const HttpRequest& Req, HttpResponse& Res,
                              NC_Environment& Env, NC_Environment& Resources)
   {
      // Trace the function call
      Trace::Tracer T(_STR_CheckParameters, _STR_ThisClass);

      if (Misc::CheckFieldExistence(Req, NVP_merchant_rn, merchantRefNum, false)==false)
      return false;

      debug << indent << " - CheckParams merchantRefNum" << merchantRefNum << endl;

      if (Misc::CheckFieldExistence(Req, NVP_url, _VAL_url, false) == false)
      return false;

      debug << indent << " - CheckParams url" << _VAL_url << endl;

      // return success if everything went OK
      return true;
   }



   // *********************************8
   // Process section
   virtual bool Process (const HttpRequest& Req, HttpResponse& Res, NC_Environment& Env)
   {
      // Trace the function call
      Trace::Tracer T(_STR_Process, _STR_ThisClass);

      // dump the nvp
      debug << indent <<
"================================================================================" << endl;
      debug << indent << " start process" << endl;

      debug << indent << " --- DumpInputStart ---" << endl;
      NameValuePairMap::Iterator I(&Req.getNVPs());
      for (I.Begin(); *I != NULL; ++I)
      {
        debug << indent << (*I)->getName().c_str() << " = ";
        debug << (*I)->getValue().c_str() << endl;
      }
      debug << indent << " --- DumpInputEnd   ---" << endl;
```

```
//=============================================================================
      // call RegisterNew command
      // Set up variables
      HttpResponse LocalRes;
      HttpRequest LocalReq;

      // Set up the LocalRequest for the RegisterNew cmd
      String queryString = "?merchant_rn=";
      queryString << merchantRefNum;
      String NVP_name;

      // Add the appropriate NVP from Req to LocalReq
      for (I.Begin(); *I != NULL; ++I)
      {
        NVP_name = (*I)->getName();
        if (NVP_name != NVP_merchant_rn && NVP_name != NVP_regtype)
        queryString << "&" << NVP_name << "=" << (*I)->getValue();
      }

   debug << "query String = " << queryString;

   LocalReq.setQUERY_STRING(queryString.c_str());


      //================================================
// get id for RegisterNew cmd

      NC_RegistrationID ID_regnew("IBM","NC","RegisterNew",1.0);

      // Use CommandManager to call RegisterNew cmd
      if( NC_CommandManager::Call(LocalReq,LocalRes,Env,ID_regnew)!=NULL ) {
           return false;
   }

      //============================================
//get current user
      User* loggeduser =
(User*)Misc::CheckEnvironmentVariable(Env,NC_Environment::_VAR_Shopper);

      DataBase& DB = *(DataBase*)Misc::CheckEnvironmentVariable(Env,
NC_Environment::_VAR_MainDatabase);

      if (loggeduser->isRegistered())
        debug << "shopper is registered: continue" << endl;
      else{
       debug << "shopper is not registered" << endl;
       return false;
      }

      //=============================================================================
      //MCUSTINFO

      MerchantUserInfoHome mcustinfo_home( DB );

      MerchantUserInfo* mcustinfo;

      String _VAL_shopref = loggeduser->getRefNum();
      String _VAL_shopgrp = "";

      mcustinfo = mcustinfo_home.Create(merchantRefNum, _VAL_shopref ,_VAL_shopgrp);
```

```
      //search for optional regtype field
      for (I.Begin(); *I != NULL; ++I)
      {
        if ((*I)->getName() == NVP_regtype){
        _VAL_regtype = (*I)->getValue();
        debug << "regtype = " << _VAL_regtype << endl;
        if (_VAL_regtype == "1" )
        mcustinfo->setField1("A");
       break;
        };
      };

      debug << indent << "Insert row in mcustinfo" << endl;

      // write to table
      if (mcustinfo->Write() != ERR_DB_NO_ERROR)
      return false;



//=============================================================================================
      //get user's new registered cookie from the LocalRes
      Cookie* LocalCookie = LocalRes.getCookie("SESSION_ID");
      String _VAL_value = LocalCookie->getValue();

      // Set the user's cookie in the wrapper cmd response
      Res.AddCookie("SESSION_ID",_VAL_value);


//=============================================================================================
      // redirect to url
      Res.setLocation(_VAL_url,false);

      debug << indent << " - Process end " << endl;
      debug << indent << "==============================================" << endl;

      // destroy objects
      mcustinfo_home.Destroy(mcustinfo);

      return true;
   }

};



// Tracing variable to identify calls to functions for your class.

const ClassName regWrapper::_STR_ThisClass("regWrapper");


// Automatically instanciate your command and registers it to the name you give it.
// "Vendor" is the name of the company you write this command for.
// "Product" is the name of the product/package this command will be part of
// "CommandName" is the actual name of your command
// 1.0 is the version of this command
static bool X1 = NC_CommandManager::Register("IBM"     ,
                                             "NC"    ,
                                             "regWrapper",
                                             1.0          ,
                                             new regWrapper
                                            );
//=============================================================================================
```

B.2 RegWrapper makefile for WinNT

```
NC_ROOT            = d:\ibm\netcommerce3

PCH                = objects\objects.pch
INC                = /I$(NC_ROOT)\adt\include
LIBS               = $(NC_ROOT)\adt\lib\nc3_containers.lib    \
                     $(NC_ROOT)\adt\lib\nc3_messages.lib      \
                     $(NC_ROOT)\adt\lib\nc3_common.lib        \
                     $(NC_ROOT)\adt\lib\nc3_dbc.lib           \
                     $(NC_ROOT)\adt\lib\nc3_pay_objects.lib   \
                     $(NC_ROOT)\adt\lib\server_objs.lib
GCFLAGS            = /nologo /c /GX $(DEBUG_COMPILE) /MD /W1 /D_X86=1 /D_X86_ \
                     /D_CONSOLE /DWIN32 $(INC) /YX"$(PCH)" /Gi
GLFLAGS            = /nologo $(DEBUG_LINK) /DLL /machine:IX86

OBJS               = regwrapper.obj

debug:
    @nmake -f makefile.nt all "DEBUG_COMPILE=/Zi /Gm /Od /DEBUG" "DEBUG_LINK=/DEBUG"
-NOLOGO

release:
    @nmake -f makefile.nt all "DEBUG_COMPILE =/O2 /Og /Oi /Ot /Oy /Ob2 /Gs /Gf /Gy /G5"
-NOLOGO

all           : regwrapper.dll

regwrapper.dll : $(OBJS)
                 link $(OBJS) $(GLFLAGS) $(LIBS) /OUT:$(@R).dll /implib:$(@R).lib
/PDB:$(@R).pdb

regwrapper.obj : $(@R).cpp $(GLBDEP)
                 cl $(GCFLAGS) $(@R).cpp

cleanall      :
                 del *.obj *.pdb *.idb *.ilk *.exp vc4?.*

copy :
    copy regwrapper.dll d:\ibm\NetCommerce3\bin\
```

B.3 RegWrapper initialization SQLs

```
insert into CMDS (REFNUM,DLL_NAME,VENDOR,PRODUCT,NAME,VERSION,URL,EXPORT,DESCRIPTION)
VALUES (20004,'regwrapper.dll','IBM','NC','regWrapper',1.00,'regWrapper',1,'');
insert into POOL_CMD (POOL_RN,CMD_RN) VALUES (1,20004);
insert into acc_mode (refnum,cmd_refnum,ssl,protect) values (10004,20004,0,0)
```

B.4 Macro file creagru.d2w

```
%{=================================================================
== Licensed Materials - Property of IBM                          ==
```

```
==                                                                            ==
== 5697-D24                                                                   ==
==                                                                            ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved. ==
==                                                                            ==
== US Government Users Restricted Rights - Use, duplication or disclosure =
== restricted by GSA ADP Schedule Contract with IBM Corp. ==
==-----------------------------------------------------------------------%}

%define {
    queryFormName = ""
    protected_command_url = ""
    merfnbrVal = ""
    firstMerfnbr = ""
    oneVar = ""
    TD="align=left bgcolor=""#6699FF"" "
    TD2="align=left bgcolor=""#FDD7AC"" "
    TD3="align=right bgcolor=""#FDD7AC"" "
    TD4="align=center bgcolor=""#6699FF"" "
    TD5="align=center bgcolor=""#FDD7AC"" "
%}

%function(dtw_odbc) GET_SHOPERGROUP() {
    SELECT sanick, sashnbr
      FROM shaddr
     WHERE sanick='$(SESSION_ID)'
%REPORT{
    %ROW{
       @DTW_assign(OHT_MERCHANTCODE, V_SASHNBR)
<BR>

    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_MERCHANTCODE, NULL) %} :continue
        default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}

%function(dtw_odbc) StoreName() {
    SELECT MENAME
      FROM MERCHANT
     WHERE MERFNBR=$(OHT_MERCHANTCODE)

    %REPORT {
        %ROW {
       @DTW_assign(OHT_MERCHANTNAME, V_MENAME)
        %}
    %}
%}

%function(dtw_odbc) GET_CONSECUTIVE() {
    SELECT keyrfnbr, keytable, keycolumn, keymaxid
      FROM keys
     WHERE keyrfnbr=13
%REPORT{
    %ROW{
       @DTW_assign(OHT_KEYTABLE, V_KEYTABLE)
       @DTW_assign(OHT_KEYCOLUMN, V_KEYCOLUMN)
       @DTW_assign(OHT_KEYMAXID, V_KEYMAXID)
       @DTW_add(V_KEYMAXID,"1",OHT_KEYCONS)
```

```
    %}
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_MERCHANTCODE, NULL) %} :continue
        default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}

%function(dtw_odbc) GET_GROUPNAMES() {
    SELECT sgrfnbr, sgname, sgtext
      FROM shopgrp
     WHERE sgmenbr=$(OHT_MERCHANTCODE)
    %REPORT {
      <table width="100%">
      <tr><th
        $(TD)><font size=2>NAME<th
        $(TD)><font size=2>DESCRIPTION
      %ROW { <tr>
        <td $(TD2)><font size=2>$(V_sgname)
        <td $(TD2)><font size=2>$(V_sgtext)
      %}
      </table>
    %}
    %MESSAGE{
       100:{
                 <table width="100%">
               <tr><th
               $(TD)><font size=2>NAME<th
               $(TD)><font size=2>DESCRIPTION
             <tr>
               <td $(TD2)><font size=2>NO GROUPS TO SHOW
               <td $(TD2)><font size=2>NO GROUPS TO SHOW
                 %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) SAVEGROUP() {
  INSERT INTO SHOPGRP VALUES ($(OHT_KEYCONS), $(OHT_MERCHANTCODE), '$(SGNAME)',
'$(SGTEXT)', '', '', '')
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) SAVEKEY() {
UPDATE KEYS
SET KEYMAXID = $(OHT_KEYCONS) WHERE KEYRFNBR = 13
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%{==================================================%}
```

```
%{ HTML REPORT Section
%{=================================================%}

%HTML_REPORT {
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>
</HEAD>
@GET_SHOPERGROUP()
@StoreName()
<FONT SIZE=+1>
<B>
Creating Shopper Groups
<BR>
for the store $(OHT_MERCHANTNAME)
</B></FONT>
    <form method="post" action="$(SERVER)SAVEGROUP">
    <table width="60%">
    <tr>
      <td $(TD)><font size=2><b>GROUP NAME:</b>
      <td><input type="input" name="SGNAME" value="" size=50 maxlength=50>
    <tr>
      <td $(TD)><font size=2><b>DESCRIPTION</b>
      <TD COLSPAN=3 HEIGHT=&{cHt}; ALIGN=&{txt};><TEXTAREA NAME="SGTEXT" ROWS=5 COLS=50
WRAP=virtual>
      </TEXTAREA>
    </table>
  <font size=2><input type=submit value="Go Register">
<BR>
<CENTER>
<FONT SIZE=+1>
<B>
Existing Groups
<B/>
</CENTER>
@GET_GROUPNAMES()
</HTML>
%}


%{=================================================%}
%{ HTML SAVEGROUP Section
%{=================================================%}

%HTML(SAVEGROUP) {
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>
</HEAD>
@GET_SHOPERGROUP()
@StoreName()
@GET_CONSECUTIVE()
@SAVEKEY()
@SAVEGROUP()
<FONT SIZE=+1>
<B>
Creating Shopper Groups
<BR>
for the store $(OHT_MERCHANTNAME)
</B></FONT>
    <form method="post" action="$(SERVER)SAVEGROUP">
```

```
    <table width="60%">
    <tr>
      <td $(TD)><font size=2><b>GROUP NAME:</b>
      <td><input type="input" name="SGNAME:" value="" size=50 maxlength=50>
    <tr>
      <td $(TD)><font size=2><b>DESCRIPTION</b>
      <TD COLSPAN=3 HEIGHT=&{cHt}; ALIGN=&{txt};><TEXTAREA NAME="SGTEXT" ROWS=5 COLS=50
WRAP=virtual>
      </TEXTAREA>
    </table>
  <font size=2><input type=submit value="Go Register">
<BR>
<CENTER>
<FONT SIZE=+1>
<B>
Existing Groups
<B/>
</CENTER>
@GET_GROUPNAMES()
</HTML>
%}
```

B.5 Macro file addshgru.d2w

```
%{=========================================================================
== Licensed Materials - Property of IBM                                  ==
==                                                                       ==
== 5697-D24                                                              ==
==                                                                       ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved. ==
==                                                                       ==
== US Government Users Restricted Rights - Use, duplication or disclosure =
== restricted by GSA ADP Schedule Contract with IBM Corp. ==
==-------------------------------------------------------------------------%}
%define {
    queryFormName = ""
    protected_command_url = ""
    merfnbrVal = ""
    firstMerfnbr = ""
    oneVar = ""
    TD="align=left bgcolor=""#6699FF"" "
    TD2="align=left bgcolor=""#FDD7AC"" "
    TD3="align=right bgcolor=""#FDD7AC"" "
    TD4="align=center bgcolor=""#6699FF"" "
    TD5="align=center bgcolor=""#FDD7AC"" "
%}

%function(dtw_odbc) GET_SHOPERGROUP() {
    SELECT sanick, sashnbr
      FROM shaddr
     WHERE sanick='$(SESSION_ID)'
%REPORT{
    %ROW{
       @DTW_assign(OHT_MERCHANTCODE, V_SASHNBR)
<BR>

    %}
```

```
%}
 %MESSAGE{
    100:{ @DTW_ASSIGN(OHT_MERCHANTCODE, NULL) %} :continue
       default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}

%function(dtw_odbc) GroupName() {
    SELECT SGNAME
      FROM SHOPGRP
     WHERE SGRFNBR=$(CODEGROUP)

    %REPORT {
        %ROW {
       @DTW_assign(GROUPNAME, V_SGNAME)
        %}
    %}
%}

%function(dtw_odbc) StoreName() {
    SELECT MENAME
      FROM MERCHANT
     WHERE MERFNBR=$(OHT_MERCHANTCODE)

    %REPORT {
        %ROW {
       @DTW_assign(OHT_MERCHANTNAME, V_MENAME)
        %}
    %}
%}

%function(dtw_odbc) GET_GROUPNAMES() {
    SELECT sgrfnbr, sgname, sgtext
      FROM shopgrp
     WHERE sgmenbr=$(OHT_MERCHANTCODE)
    %REPORT {
      <table width="100%">
      <tr><th
        $(TD)><font size=2>NAME<th
        $(TD)><font size=2>DESCRIPTION
      %ROW { <tr>
        <td $(TD2)><font size=2><a
href="$(SERVER)ADDTOGROUP?CODEGROUP=$(V_sgrfnbr)&CODESHOPP=$(CODESHOPP)">$(V_sgname)</a>
        <td $(TD2)><font size=2>$(V_sgtext)
      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) GET_SHOPPNAMES() {
    SELECT shrfnbr, shlogid, shshtyp,
           sarfnbr, sashnbr, sanick, satitle, safname, samname, salname, saemail1
      FROM SHOPPER, SHADDR
     WHERE SHSHTYP = 'R'
       AND (shrfnbr in (select mcshnbr from MCUSTINFO WHERE MCSGNBR is NULL AND MCMENBR =
$(OHT_MERCHANTCODE)))
```

```
       AND SHRFNBR = SASHNBR
       AND SHLOGID = SANICK
  ORDER BY safname
    %REPORT {
      <CAPTION>
    Chose a shopper from the next list
      </CAPTION>
      <table width="100%">
      <tr><th
        $(TD)><font size=2>CODE<th
        $(TD)><font size=2>NAME<th
        $(TD)><font size=2>E-MAIL
      %ROW { <tr>
      @DTW_CONCAT($(V_satitle)," ",OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME)," ",OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME),$(V_safname),OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME)," ",OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME),$(V_salname),OHT_REALNAME)
        <td $(TD2)><font size=2><a
href="$(SERVER)SHOWGROUPS?CODESHOPP=$(V_sashnbr)">$(V_sashnbr)</a>
        <td $(TD2)><font size=2>$(OHT_REALNAME)
        <td $(TD2)><font size=2>$(V_saemail1)

      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR $(OHT_MERCHANTCODE)%} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) GET_SHOPPCHOOS() {
    SELECT sarfnbr, sashnbr, sanick, satitle, safname, samname, salname, saemail1
      FROM SHADDR
     WHERE SASHNBR = $(CODESHOPP)
  ORDER BY safname
    %REPORT {
      <CAPTION>
    <FONT SIZE=+1>
    <B>
    The choosen shopper was
    </B></FONT>
    <BR>
      </CAPTION>
      <table width="100%">
      <tr><th
        $(TD)><font size=2>CODE<th
        $(TD)><font size=2>NAME<th
        $(TD)><font size=2>E-MAIL
      %ROW { <tr>
      @DTW_CONCAT($(V_satitle)," ",OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME)," ",OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME),$(V_safname),OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME)," ",OHT_REALNAME)
      @DTW_CONCAT($(OHT_REALNAME),$(V_salname),OHT_REALNAME)
        <td $(TD2)><font size=2>$(V_sashnbr)
        <td $(TD2)><font size=2>$(OHT_REALNAME)
        <td $(TD2)><font size=2>$(V_saemail1)

      %}
```

```
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) ADD_SHOPTOGROUP() {
UPDATE MCUSTINFO
   SET MCSGNBR = $(CODEGROUP)
 WHERE MCSHNBR = $(CODESHOPP) AND MCMENBR = $(OHT_MERCHANTCODE)
    %MESSAGE{
       100:{ ADD_SHOPTOGROUP() %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) SHOW_STATUS() {
    SELECT mcshnbr,
           sarfnbr, sashnbr, sanick, satitle, safname, samname, salname, saemail1
      FROM MCUSTINFO, SHADDR
     WHERE mcshnbr = SASHNBR
       AND MCMENBR = $(OHT_MERCHANTCODE)
       AND MCSGNBR = $(CODEGROUP)
       AND SAADRFLG = 'P'
  ORDER BY safname
    %REPORT {
      <table width="100%">
      <tr><th
        $(TD)><font size=2>CODE<th
        $(TD)><font size=2>NAME<th
        $(TD)><font size=2>E-MAIL
      %ROW { <tr>
     @DTW_CONCAT($(V_satitle)," ",OHT_REALNAME)
     @DTW_CONCAT($(OHT_REALNAME)," ",OHT_REALNAME)
     @DTW_CONCAT($(OHT_REALNAME),$(V_safname),OHT_REALNAME)
     @DTW_CONCAT($(OHT_REALNAME)," ",OHT_REALNAME)
     @DTW_CONCAT($(OHT_REALNAME),$(V_salname),OHT_REALNAME)
        <td $(TD2)><font size=2>$(V_sashnbr)
        <td $(TD2)><font size=2>$(OHT_REALNAME)
        <td $(TD2)><font size=2>$(V_saemail1)

      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%{=================================================%}
%{ HTML Report Section
%{=================================================%}

%HTML_REPORT {
```

```
<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>

@GET_SHOPERGROUP()
@StoreName()
<FONT SIZE=+1>
<B>
Adding Registered Shoppers to
<BR>
Shopper Groups
</B></FONT>
<BR>
@GET_SHOPPNAMES()
</HTML>

%}


%{=================================================%}
%{ HTML SHOWGROUPS Section
%{=================================================%}

%HTML(SHOWGROUPS) {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPPCHOOS()
@GET_SHOPERGROUP()
<FONT SIZE=+1>
<B>
Now pick a Shopper Group from the
<BR>
Shopper Groups list
</B></FONT>
<BR>
@GET_GROUPNAMES()
</HTML>

%}


%{=================================================%}
%{ HTML ADDTOGROUP Section
%{=================================================%}

%HTML(ADDTOGROUP) {

<HTML>

<HEAD>
```

```
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPERGROUP()
@StoreName()
@GroupName()
@ADD_SHOPTOGROUP()
<FONT SIZE=+1>
<B>
The shoppers whom belongs to the
<BR>
Shopper Group "$(GROUPNAME)" are
</B></FONT>
<BR>
@SHOW_STATUS()
</HTML>

%}
```

B.6 Macro addprgru.d2w

```
%{============================================================================
== Licensed Materials - Property of IBM                                     ==
==                                                                          ==
== 5697-D24                                                                 ==
==                                                                          ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved. ==
==                                                                          ==
== US Government Users Restricted Rights - Use, duplication or disclosure =
== restricted by GSA ADP Schedule Contract with IBM Corp. ==
==----------------------------------------------------------------------------%}

%define {
    queryFormName = ""
    protected_command_url = ""
    merfnbrVal = ""
    firstMerfnbr = ""
    oneVar = ""
    TD="align=left bgcolor=""#6699FF"" "
    TD2="align=left bgcolor=""#FDD7AC"" "
    TD3="align=right bgcolor=""#FDD7AC"" "
    TD4="align=center bgcolor=""#6699FF"" "
    TD5="align=center bgcolor=""#FDD7AC"" "
%}

%function(dtw_odbc) GET_SHOPERGROUP() {
    SELECT sanick, sashnbr
      FROM shaddr
     WHERE sanick='$(SESSION_ID)'
%REPORT{
    %ROW{
       @DTW_assign(OHT_MERCHANTCODE, V_SASHNBR)
<BR>

    %}
%}
```

```
 %MESSAGE{
    100:{ NO HAY NADA QUE PRESENTAR %} :continue
        default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue

 %}
%}

%function(dtw_odbc) GroupName() {
    SELECT SGRFNBR,SGNAME
      FROM SHOPGRP
     WHERE SGRFNBR=$(CODEGROUP)

    %REPORT {
        %ROW {
       @DTW_assign(GROUPNAME, V_SGNAME)
        %}
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GroupName() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) ProduName() {
    SELECT PRRFNBR, PRSDESC
      FROM PRODUCT
     WHERE PRRFNBR=$(CODEPROD)

    %REPORT {
        %ROW {
       @DTW_assign(PRODUNAME, V_PRSDESC)
        %}
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GroupName() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) StoreName() {
    SELECT MENAME, MECUR
      FROM MERCHANT
     WHERE MERFNBR=$(OHT_MERCHANTCODE)

    %REPORT {
        %ROW {
       @DTW_assign(OHT_MERCHANTNAME, V_MENAME)
       @DTW_assign(OHT_MERCHANTCUR, V_MECUR)
        %}
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with StoreName() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) GET_GROUPNAMES() {
    SELECT sgrfnbr, sgname, sgtext
      FROM shopgrp
```

```
     WHERE SGMENBR = $(OHT_MERCHANTCODE)
       AND (sgrfnbr not in (SELECT PPSGNBR FROM PRODPRCS WHERE PPMENBR =
$(OHT_MERCHANTCODE) AND PPPRNBR = $(CODEPROD) AND PPSGNBR is not NULL))
    %REPORT {
      <table width="100%">
      <tr><th
        $(TD4)><font size=2>NAME<th
        $(TD4)><font size=2>DESCRIPTION
      %ROW { <tr>
        <td $(TD2)><font size=2><a
href="$(SERVER)INPUTPRICE?CODEGROUP=$(V_sgrfnbr)&CODEPROD=$(CODEPROD)">$(V_sgname)</a>
        <td $(TD2)><font size=2>$(V_sgtext)
      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) GET_PRODUNAMES() {
    SELECT PRNBR, PRMENBR, PRRFNBR, PRSDESC, PRFULL, PRPUB
      FROM PRODUCT
     WHERE PRMENBR=$(OHT_MERCHANTCODE)
       AND PRPUB=1
  ORDER BY PRNBR
    %REPORT {
      <CAPTION>
    Chose a product from the next list
      </CAPTION>
      <table width="100%">
      <tr><th
        $(TD4)><font size=2>SKU<th
        $(TD4)><font size=2>DESCRIPTION<th
        $(TD4)><font size=2>IMAGE
      %ROW { <tr>
        <td $(TD2)><font size=2><a
href="$(SERVER)SHOWPRICES?CODEPROD=$(V_prrfnbr)">$(V_prnbr)</a>
        <td $(TD2)><font size=2>$(V_PRSDESC)
        <td $(TD5)><font size=2><IMG SRC="$(V_prfull)" ALT="$(V_prsdesc)" HEIGHT=100>

      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_SHOPERGROUP() in file $(DTW_MACRO_FILENAME) because
of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) SHOWPRICES() {
SELECT PPSGNBR, PPPRC,
       SGNAME, SGRFNBR
  FROM PRODPRCS, SHOPGRP
 WHERE PRODPRCS.PPMENBR=$(OHT_MERCHANTCODE)
   AND PRODPRCS.PPPRNBR=$(CODEPROD)
   AND PRODPRCS.PPSGNBR=SHOPGRP.SGRFNBR
    %REPORT {
      <table width="100%">
      <tr><th
```

```
        $(TD4)><font size=2>SHOPPER GROUP<th
        $(TD4)><font size=2>PRICE
      %ROW { <tr>
        <td $(TD2)><font size=2><a
href="$(SERVER)REMOVEPRICE?CODEPROD=$(CODEPROD)&CODEGROUP=$(V_SGRFNBR)&OHT_MERCHANTCODE=
$(OHT_MERCHANTCODE)">$(V_SGNAME)</a>
        <td $(TD5)><font size=2><a
href="$(SERVER)EDITPRICES?CODEPROD=$(CODEPROD)&CODEGROUP=$(V_SGRFNBR)&OHT_MERCHANTCODE=$
(OHT_MERCHANTCODE)">$(V_PPPRC)</a>
      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with SHOWPRICES() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) GET_PRODUCHOOS() {
    SELECT PRNBR, PRMENBR, PRRFNBR, PRSDESC, PRFULL, PRPUB,
           PPMENBR, PPPRNBR, PPPRC
      FROM PRODUCT, PRODPRCS
     WHERE PRRFNBR=$(CODEPROD)
       AND PPMENBR=$(OHT_MERCHANTCODE)
       AND PPPRNBR=$(CODEPROD)
       AND PPSGNBR is NULL
  ORDER BY PRRFNBR
    %REPORT {
      <table width="60%" border="0">
      %ROW { <tr>
        <b>$(V_prnbr)</b>
        <BR>
        $(V_PRSDESC)
        <BR>
        <b>PRICE:</b>$ $(V_PPPRC)
        <BR>
        <IMG SRC="$(V_prfull)" ALT="$(V_prsdesc)" HEIGHT=100>
      %}
      </table>
    %}
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with GET_PRODUCHOOS $(DTW_MACRO_FILENAME) because of error
$(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) GET_CONSECUTIVE() {
    SELECT keyrfnbr, keytable, keycolumn, keymaxid
      FROM keys
     WHERE keyrfnbr=15
     %REPORT{
       @DTW_assign(OHT_KEYTABLE, V_KEYTABLE)
       @DTW_assign(OHT_KEYCOLUMN, V_KEYCOLUMN)
       @DTW_assign(OHT_KEYMAXID, V_KEYMAXID)
       @DTW_add(V_KEYMAXID,"1",OHT_KEYCONS)
     %}
     %MESSAGE{
         100:{ NO HAY NADA QUE PRESENTAR %} :continue
           default:{<BR>Problem with GET_CONSECUTIVE() in file $(DTW_MACRO_FILENAME)
because of error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
```

```
 %}
%}


%function(dtw_odbc) SAVEKEY() {
UPDATE KEYS
SET KEYMAXID = $(OHT_KEYCONS) WHERE KEYRFNBR = 15
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with SAVEKEY() in file $(DTW_MACRO_FILENAME) because of error
$(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) SAVEPRICE() {
  INSERT INTO PRODPRCS VALUES ($(OHT_KEYCONS), $(OHT_MERCHANTCODE), $(CODEPROD),
$(CODEGROUP), $(PPPRICE), '$(OHT_MERCHANTCUR)', 1,
'2000-03-24-14.49.48.000000','9999-12-31-24.00.00.000000','','','')
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with SAVEPRICE() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}

%function(dtw_odbc) EDITPRICES() {
SELECT PPRFNBR, PPSGNBR, PPPRC,
       SGNAME, SGRFNBR
  FROM PRODPRCS, SHOPGRP
 WHERE PRODPRCS.PPMENBR=$(OHT_MERCHANTCODE)
   AND PRODPRCS.PPPRNBR=$(CODEPROD)
   AND PRODPRCS.PPSGNBR=$(CODEGROUP)
     %REPORT{
       <form method="post"
action="$(SERVER)CHANGEPRICES?CODEGROUP=$(CODEGROUP)&CODEPROD=$(CODEPROD)">
       <table width="60%">
         <tr>
         <td $(TD)><font size=2><b>GROUP NAME:</b>
         <td $(TD2)><font size=2><b>$(GROUPNAME)</b>
         <tr>
         <td $(TD)><font size=2><b>PRICE:</b>
         <td $(TD2)><input type="input" name="PPPRICE" value=$(V_PPPRC) size=15
maxlength=15>
         <input type=hidden name=CODEGROUP value=$(CODEGROUP)>
         <input type=hidden name=CODEPROD value=$(CODEPROD)>
         <input type=hidden name=CODEPRICE value=$(V_PPRFNBR)>
       </table>
       <font size=2><input type=submit value="Go Change">
     %}
     %MESSAGE{
         100:{ NO HAY NADA QUE PRESENTAR %} :continue
           default:{<BR>Problem with EDITPRICE() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
 %}
%}


%function(dtw_odbc) CHANGEPRICES() {
UPDATE PRODPRCS
SET PPPRC = $(PPPRICE) WHERE PPRFNBR = $(CODEPRICE)
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
```

```
      default:{<BR>Problem with CHANGEPRICES() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%function(dtw_odbc) REMOVEPRICE() {
DELETE FROM PRODPRCS
 WHERE PRODPRCS.PPMENBR=$(OHT_MERCHANTCODE)
   AND PRODPRCS.PPPRNBR=$(CODEPROD)
   AND PRODPRCS.PPSGNBR=$(CODEGROUP)
    %MESSAGE{
       100:{ NO HAY NADA QUE PRESENTAR %} :continue
      default:{<BR>Problem with REMOVEPRICE() in file $(DTW_MACRO_FILENAME) because of
error $(RETURN_CODE). <BR>$(DTW_DEFAULT_MESSAGE)%} :continue
    %}
%}


%{=================================================%}
%{ HTML Report Section
%{=================================================%}

%HTML_REPORT {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>

@GET_SHOPERGROUP()
@StoreName()
<FONT SIZE=+1>
<B>
Assigning Product Prices to
<BR>
Shopper Groups
</B></FONT>
<BR>
@GET_PRODUNAMES()
</HTML>

%}


%{=================================================%}
%{ HTML SHOWPRICES Section
%{=================================================%}

%HTML(SHOWPRICES) {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
```

```
@GET_SHOPERGROUP()
@StoreName()
@GET_PRODUCHOOS()
@SHOWPRICES()
<P>
<CENTER>
<FONT SIZE=+1>
<B>
Available Shopper Groups
</B></FONT>
</CENTER>
@GET_GROUPNAMES()
</HTML>

%}


%{=================================================%}
%{ HTML INPUTPRICE Section
%{=================================================%}

%HTML(INPUTPRICE) {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPERGROUP()
@GroupName()
@ProduName()
@StoreName()
@GET_PRODUCHOOS()
@SHOWPRICES()
<BR>
<form method="post"
action="$(SERVER)SAVEPRICE?CODEGROUP=$(CODEGROUP)&CODEPROD=$(CODEPROD)">
<table width="60%">
  <tr>
  <td $(TD)><font size=2><b>GROUP NAME:</b>
  <td $(TD2)><font size=2><b>$(GROUPNAME)</b>
  <tr>
  <td $(TD)><font size=2><b>PRICE:</b>
  <td $(TD2)><input type="input" name="PPPRICE" value=0.00 size=15 maxlength=15>
  <input type=hidden name=CODEGROUP value=$(CODEGROUP)>
  <input type=hidden name=CODEPROD value=$(CODEPROD)>
</table>
<font size=2><input type=submit value="Go Register">
</HTML>

%}


%{=================================================%}
%{ HTML SAVEPRICE Section
%{=================================================%}

%HTML(SAVEPRICE) {

<HTML>
```

```
<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPERGROUP()
@GroupName()
@StoreName()
@GET_CONSECUTIVE()
@SAVEKEY()
@SAVEPRICE()
@GET_PRODUCHOOS()
@SHOWPRICES()
<P>
<CENTER>
<FONT SIZE=+1>
<B>
Available Shopper Groups
</B></FONT>
</CENTER>
@GET_GROUPNAMES()
</HTML>

%}


%{=================================================%}
%{ HTML EDITPRICES Section
%{=================================================%}

%HTML(EDITPRICES) {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPERGROUP()
@GroupName()
@ProduName()
@StoreName()
@GET_PRODUCHOOS()
@SHOWPRICES()
<BR>
@EDITPRICES()
</HTML>

%}


%{=================================================%}
%{ HTML CHANGEPRICES Section
%{=================================================%}

%HTML(CHANGEPRICES) {

<HTML>
```

```
<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPERGROUP()
@GroupName()
@ProduName()
@StoreName()
@CHANGEPRICES()
@GET_PRODUCHOOS()
@SHOWPRICES()
<BR>
<P>
<CENTER>
<FONT SIZE=+1>
<B>
Available Shopper Groups
</B></FONT>
</CENTER>
@GET_GROUPNAMES()
</HTML>

%}


%{=================================================%}
%{ HTML REMOVEPRICE Section
%{=================================================%}

%HTML(REMOVEPRICE) {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>Shopper Groups</TITLE>

</HEAD>
@GET_SHOPERGROUP()
@GroupName()
@ProduName()
@StoreName()
@REMOVEPRICE()
@GET_PRODUCHOOS()
@SHOWPRICES()
<BR>
<P>
<CENTER>
<FONT SIZE=+1>
<B>
Available Shopper Groups
</B></FONT>
</CENTER>
@GET_GROUPNAMES()
</HTML>

%}
```

Appendix C. Source code for cross and up selling

This appendix contains program source code listing for relating products in cross and up selling as used in Chapter 6, “Cross-sell and up-sell” on page 103.

C.1 productRel.d2w Net.Data macro to associate products

```
%{==========================================================================
== Licensed Materials - Property of IBM                                   ==
==                                                                        ==
== 5697-D24                                                               ==
==                                                                        ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.            ==
==                                                                        ==
== US Government Users Restricted Rights - Use, duplication or disclosure ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                 ==
==                                                                        ==
==*** This macro takes the MERFNBR as the input parameter named 'merfnbr'***==
==                                                                        ==
==*** and retrieves product informations for this store ******************==
==========================================================================%}
%DEFINE
{
  DATABASE="demomall"
  LOGIN="db2inst1"
  PASSWORD="chuy5"
  DTW_HTML_TABLE="YES"
  plmenbr=""
%}

%FUNCTION (DTW_ODBC) updateDb()
{
 insert into prprrel values( ((select max(plrfnbr) from prprrel) +
1),$(plmenbr),$(plprtnbr),$(plprnbr),$(plrelprnbr),$(plqty),null,null,$(plpub),null,null
)
%}

%FUNCTION (DTW_ODBC) selectProductNum()
{
   select prrfnbr,prnbr,prsdesc,ppprc from product,prodprcs where prmenbr=$(merfnbr) and
prrfnbr=ppprnbr and ppmenbr = $(merfnbr) order by prrfnbr
   %REPORT
   {
     %ROW
     {
    <OPTION VALUE="$(V_prrfnbr)"> $(V_prrfnbr)- $(V_prnbr)-$(V_prsdesc)-$ $(V_ppprc)
    </OPTION>
     %}
   %}
%}

%FUNCTION (DTW_ODBC) selectRelated()
{
    select plprnbr,plrelprnbr,prtname from prprrel,prreltype where plmenbr=$(plmenbr) and
plprtnbr = prtrfnbr
    %REPORT
    {
```

```
        <TABLE BORDER CELLPADDING = 2>
    <TR><TH> PRODUCT NO. </TH> <TH> RELATED PROD NO. </TH> <TH> RELATION TYPE </TH> </TR>
    %ROW
    {
      <TR>
        <TD> $(V_plprnbr) </TD>
        <TD> $(V_plrelprnbr) </TD>
        <TD> $(V_prtname) </TD>
      </TR>
    %}
        </TABLE>
     %}
%}

%HTML_REPORT
{
 <HTML>
  <HEAD> <TITLE> product Catalog Relationship editor </TITLE>
  </HEAD>

  <BODY>
   <FORM METHOD="GET" ACTION="PROCESS">
   @DTW_ASSIGN(plmenbr,merfnbr)
   <INPUT TYPE="HIDDEN" NAME="plmenbr" VALUE="$(plmenbr)">
   <P>
   Select the first product      : <SELECT NAME="plprnbr">
   @selectProductNum()
   </SELECT>
<P>
   Select the related product : <SELECT NAME="plrelprnbr">
   @selectProductNum()
   </SELECT>
<p>

Select the relationship type : <INPUT TYPE="RADIO" NAME="plprtnbr" VALUE="2" CHECKED>
Up-Sell
   <INPUT TYPE="RADIO" NAME="plprtnbr" VALUE="3"> Cross-Sell
   <INPUT TYPE="RADIO" NAME="plprtnbr" VALUE="1"> Compatible
<p>
   Enter the related product quantity : <INPUT TYPE="TEXT" size="2" VALUE="1"
NAME="plqty">
<p>
   Select whether to publish the relation: <INPUT TYPE="RADIO" NAME="plpub" VALUE="1"
CHECKED> Publish
<INPUT TYPE="RADIO" NAME="plpub" VALUE="2"> Not to Publish

   <BR>
   <INPUT TYPE="SUBMIT" VALUE="Relate">
  </BODY>

 </HTML>
%}

%HTML(PROCESS)
{
 <HTML>
  <HEAD>
  </HEAD>
   <BODY>
       @updateDb()
       You have related the following products.
       @selectRelated()
   </BODY>
```

```
 </HTML>
%}
```

C.2 Customized product display (cat_product.d2w) macro

```
%{=============================================================================
== Licensed Materials - Property of IBM                                      ==
==                                                                           ==
== 5697-D24                                                                  ==
==                                                                           ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.               ==
==                                                                           ==
== US Government Users Restricted Rights - Use, duplication or disclosure    ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                    ==
==                                                                           ==
==*** This macro takes the MERFNBR as the input parameter named 'merfnbr'***==
==                                                                           ==
==*** and retrieves product informations for this store ********************==
=============================================================================%}


%include "7286/include.inc"
%include "7286/theme.inc"

%INCLUDE "/ClassicStoreModel/translation_text.inc"

%define {
    SHOWSQL="NO"
    NC_NOW = "{fn now()}"
    prrelprnbr=""
    sellcnt=""
%}

%INCLUDE "/ClassicStoreModel/service_info.inc"


%{  Cross Sell, Up sell Function Definitions   %}

%function(dtw_odbc) CountSell(IN l_prtname){
  select count(plrelprnbr) as sellcnt
  from prprrel, prreltype, product
  where
  plpub = 1 and
  prpub = 1 and
  prrfnbr = plrelprnbr and
  plprnbr = $(prrfnbr) and
  plprtnbr = prtrfnbr and
  prtname = '$(l_prtname)'


  %REPORT{
   %ROW{
    @DTW_ASSIGN(sellcnt,V_sellcnt)
   %}
  %}
  %MESSAGE{100:{ %} :continue %}
%}

%function(dtw_odbc) Sell(IN l_prtname){
  select plrelprnbr, prrfnbr, prnbr, prsdesc
```

```
  from prprrel, prreltype, product
  where
  plpub = 1 and
  prpub = 1 and
  prrfnbr = plrelprnbr and
  plprnbr = $(prrfnbr) and
  plprtnbr = prtrfnbr and
  prtname = '$(l_prtname)'


  %REPORT{
   %ROW{
       @DTW_ASSIGN(prrelprnbr,$(V_prrfnbr))
     <tr>
 <td><a href=#
onClick=ShowSell($(prrelprnbr))>$(V_prnbr)</a> </td><td>$(V_prsdesc)</td> <br>
     </tr>
   %}
  %}
  %MESSAGE{100:{ %} :continue %}
%}

%{ Other Function definitions %}

%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{

<HTML>

@GET_STORE_SERVICELEVEL()
@GET_PRODUCTNAME()
@GET_DISTINCTATTRIBUTES()

<HEAD>
    <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1999 01:01:01 GMT">

    <SCRIPT LANGUAGE="javascript">
        function ShowPriceList() {
        window.open("/cgi-bin/ncommerce3/ExecMacro/" + $(DirectoryName) +
"/cat_prodpricelist.d2w/report?product_rn=" +
$(prrfnbr),"pricelist$(prrfnbr)","toolbar=no,location=no,directories=no,status=no,menuba
r=no,resizable=yes,width=300,height=500,prompt=no,scrollbars=1,setTimeOut=no");
        }
    </SCRIPT>

</HEAD>

<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)" TEXT="$(TextCol)"
LINK="$(LinkCol)" VLINK="$(VLinkCol)" ALINK="$(ALinkCol)">

@DTW_ADD(MerchantRefNum, "0", "2" , SecurityCheck)
@DTW_ADD(prrfnbr, "0", "2" , SecurityCheck)
@DTW_ADD(prmenbr, "0", "2" , SecurityCheck)

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <TR>
        <TD ALIGN=$(TitleAlignment) VALIGN="center">
```

```
          <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(TXT_TITLE_CATALOG)</B></FONT>
        </TD>
    </TR>
</TABLE>
</CENTER>

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

<TR>
    <TD align="left">
        <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRY)</FONT>
    </TD>
</TR>

</TABLE>
</CENTER>

<BR>

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <TR>
        <TD ALIGN=$(TitleAlignment) VALIGN="center">
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(PRODUCTNAME)</B></FONT>
        </TD>
    </TR>



    %IF (REFERENCEPAGE != "CATALOG" && REFERENCEPAGE != "SEARCH")

        <TR><TD><BR></TD></TR>
        <TR>
            <TD align="left">
                <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>($(TXT_TITLE_PRODUCT))</B></FONT>
            </TD>
        </TR>
    %ENDIF

</TABLE>
</CENTER>

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <!--A
HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/InterestItemAdd?product_rn=$(prrfnbr)&merch
ant_rn=$(MerchantRefNum)&WISHLISTSHOPPER_RN=$(SESSION_RN)&url=http://$(HOST_NAME)/cgi-bi
n/ncommerce3/InterestItemDisplay">
    <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)">Add to my wishlist</FONT>
    </A-->

    @DISPLAY_PRODUCT_IMAGE()
    @DISPLAY_PRODUCT_INFO()
    @DISPLAY_PRODATTR()

<!-- code added for cross sell start -->
```

```
@CountSell("Up-sell")

%IF(sellcnt > "0")
    <tr>
        <th colspan="2" align="left">
        This is better (Upsell) ...:
        </th>
    </tr>

    @Sell("Up-sell")

%ENDIF

@CountSell("Cross-sell")

%IF(sellcnt > "0" )
    <tr>
        <th colspan="2" align="left">
        May we suggest (Cross-sell) ...:
        </th>
    </tr>

    @Sell("Cross-sell")
%ENDIF

@CountSell("Compatible")

%IF(sellcnt > "0")
    <tr>
        <th colspan="2" align="left">
        Compatible accessories:
        </th>
    </tr>

         @Sell("Compatible")
%ENDIF

<SCRIPT LANGUAGE="javascript">

   function ShowSell(prrelprnbr)
   {

window.open("http://$(HOST_NAME)/cgi-bin/ncommerce3/ExecMacro/product/$(prmenbr)/cat_pro
duct_win.d2w/report?prrfnbr="+prrelprnbr+"&prmenbr=$(prmenbr)","sellwindow","toolbar=no,
location=no,directories=no,status=no,menubar=no,resizable=yes,width300,height=500,prompt
=no,scrollbars=1,setTimeOut=no");
   }
   </SCRIPT>

<!-- code added for cross sell end  -->


    %IF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor == NULL || ConvDivideFactor ==
NULL))

        @DISPLAY_PRODUCT_SINGLEPRICE()

    %ELIF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor != NULL && ConvDivideFactor !=
NULL))

        @DISPLAY_PRODUCT_DUALPRICE()
```

```
    %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor == NULL || ConvDivideFactor ==
NULL))

        @DISPLAY_PRODUCT_SINGLEPRICE_RANGE()
        @DISPLAY_ITEMS_DROPDOWN()

    %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor != NULL && ConvDivideFactor !=
NULL))

        @DISPLAY_PRODUCT_DUALPRICE_RANGE()
        @DISPLAY_ITEMS_DROPDOWN()

    %ENDIF

    <TR><TD ALIGN="center"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSizeSmall)">$(MSG_REMOVEFROMSHOPCART)</FONT></TD></TR>

</TABLE>

</BODY>
</HTML>

%}
```

C.3 Definition file of mass import utility (prprrel.ini)

```
#VERSION;32

#TABLE;shopgrp
#COLUMNS;sgmenbr;sgname
#CONSTTOKEN;sgmenbr;merchant

#TABLE;prspcode
#COLUMNS;psrfnbr;psmenbr;pscode;psspmthd
#CONSTTOKEN;psmenbr;merchant

#TABLE;category
#COLUMNS;cgrfnbr;cgmenbr;cgname;cgnbr;cgsdesc;cgldesc;cgdisplay;cgthmb;cgfull;cgpub;cgfi
eld1;cgfield2;cgstmp
#UI;cgmenbr;cgnbr
#CONSTTOKEN;cgmenbr;merchant
#REFNUM;cgrfnbr
#DEFAULT;cgpub;1
#DEFAULT;cgstmp;currenttime

#TABLE;cgryrel
#COLUMNS;crmenbr;crpcgnbr;crccgnbr;crseqnbr
#UI;crmenbr;crpcgnbr;crccgnbr
#CONSTTOKEN;crmenbr;merchant
#FORREF;cgryrel;crpcgnbr;category;cgrfnbr;cgmenbr;cgnbr
#FORREF;cgryrel;crccgnbr;category;cgrfnbr;cgmenbr;cgnbr

#TABLE;catesgp
#COLUMNS;csmenbr;cscgnbr;cssgnbr;csdisplay;csdesc;csfield1;csfield2
#UI;csmenbr;cscgnbr;cssgnbr
#CONSTTOKEN;csmenbr;merchant
#FORREF;catesgp;cscgnbr;category;cgrfnbr;cgmenbr;cgnbr
#FORREF;catesgp;cssgnbr;shopgrp;sgrfnbr;sgmenbr;sgname

#TABLE;product
```

```
#COLUMNS;prrfnbr;prmenbr;prnbr;prprfnbr;prsdesc;prldesc1;prldesc2;prldesc3;prthmb;prfull
;prpub;prwght;prwmeas;prlngth;prwidth;prheight;prsmeas;prpsnbr;prpcode;prurl;prvent;prav
date;prspecial;prstmp;prfield1;prfield2;prfield3;prfield4;prfield5;prdconbr
#UI;prmenbr;prnbr
#CONSTTOKEN;prmenbr;merchant
#FORREF;product;prprfnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;product;prpsnbr;prspcode;psrfnbr;psmenbr;pscode;psspmthd
#REFNUM;prrfnbr
#DEFAULT;prpub;1
#DEFAULT;prwght;0
#DEFAULT;prlngth;0
#DEFAULT;prwidth;0
#DEFAULT;prheight;0
#DEFAULT;prstmp;currenttime

#TABLE;prodsgp
#COLUMNS;psmenbr;psprnbr;pssgnbr;psdisplay;psdesc;psfield1;psfield2
#UI;psmenbr;psprnbr;pssgnbr
#CONSTTOKEN;psmenbr;merchant
#FORREF;prodsgp;psprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prodsgp;pssgnbr;shopgrp;sgrfnbr;sgmenbr;sgname

#TABLE;prodprcs
#COLUMNS;pprfnbr;ppmenbr;ppprnbr;ppsgnbr;ppprc;ppcur;pppre;ppdeffs;ppdefff;ppfield1;ppfi
eld2
#UI;ppmenbr;ppprnbr;ppsgnbr;ppcur;pppre
#CONSTTOKEN;ppmenbr;merchant
#FORREF;prodprcs;ppprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prodprcs;ppsgnbr;shopgrp;sgrfnbr;sgmenbr;sgname
#REFNUM;pprfnbr
#DEFAULT;pppre;0
#DEFAULT;ppdeffs;currenttime
#DEFAULT;ppdefff;9999-12-31-00.00.00

#TABLE;prodatr
#COLUMNS;pamenbr;paprnbr;paname;paval;pafield1
#UI;pamenbr;paprnbr;paname
#CONSTTOKEN;pamenbr;merchant
#FORREF;prodatr;paprnbr;product;prrfnbr;prmenbr;prnbr

#TABLE;proddstatr
#COLUMNS;pdmenbr;pdprnbr;pdname;pddesc1;pddesc2;pdseqnbr
#UI;pdmenbr;pdprnbr;pdname
#CONSTTOKEN;pdmenbr;merchant
#FORREF;proddstatr;pdprnbr;product;prrfnbr;prmenbr;prnbr

#TABLE;cgprrel
#COLUMNS;cpmenbr;cpcgnbr;cpprnbr;cpseqnbr
#UI;cpmenbr;cpcgnbr;cpprnbr
#CONSTTOKEN;cpmenbr;merchant
#FORREF;cgprrel;cpcgnbr;category;cgrfnbr;cgmenbr;cgnbr
#FORREF;cgprrel;cpprnbr;product;prrfnbr;prmenbr;prnbr

#TABLE;prreltype
#COLUMNS;prtrfnbr;prtname;prtdesc;prtfield1;prtfield2;prtfield3
#UI;prtname
#REFNUM;prtrfnbr

#TABLE;prprrel
#COLUMNS;plrfnbr;plmenbr;plpub;plqty;plprtnbr;plsgnbr;plprnbr;plseqnbr;plrelprnbr
#UI;plmenbr;plprnbr;plrelprnbr;plsgnbr;plprtnbr
#CONSTTOKEN;plmenbr;merchant
#REFNUM;plrfnbr
```

```
#FORREF;prprrel;plprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prprrel;plrelprnbr;product;prrfnbr;prmenbr;prnbr
#FORREF;prprrel;plprtnbr;prreltype;prtrfnbr;prtname
```

Appendix D. Store creation wizard sample code

This appendix contains source code listing for programs used in Chapter 8, “Customizing the store creation wizard” on page 129.

D.1 Three level category example, file more_cat.inc

```
%{=============================================================================
== Licensed Materials - Property of IBM                                      ==
==                                                                           ==
== 5697-D24                                                                  ==
==                                                                           ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.               ==
==                                                                           ==
== US Government Users Restricted Rights - Use, duplication or disclosure    ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                    ==
=============================================================================%}
%define{
catTable = %TABLE(20)
DTW_SET_TOTAL_ROWS="YES"
DTW_DEFAULT_REPORT = "NO"
%}


%function(dtw_odbc) GET_CHILD_CATS(IN newcgrfnbr){

SELECT cgrfnbr, cgname, cgldesc
FROM   category, cgryrel
WHERE  crccgnbr=cgrfnbr and crpcgnbr=$(newcgrfnbr) and crmenbr=$(cgmenbr) and cgpub=1
ORDER BY crseqnbr

    %REPORT{
        <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
        <B>@DTW_TB_rGETV(catTable,loop,"2")</B>
        </FONT>
        </td><td>------</td><td>
        %ROW{
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
            <A
HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_cgrfnbr)&cgmenb
r=$(MerchantRefNum)">
            <B>$(V_cgname)</B></A> 
            </FONT><br>
        %}
    %}
    %MESSAGE{
        100:{
        <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
        <A
HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(newcgrfnbr)&cgmen
br=$(MerchantRefNum)">
        <B>@DTW_TB_rGETV(catTable,loop,"2")</B></A>
        </FONT>
        %} :continue
        default:{%}:continue

    %}
%}
```

```
%MACRO_FUNCTION DISPLAY_CATS(){
@DTW_ASSIGN(loop,"1")
@DTW_TB_ROWS(catTable,totalrows)
%IF (totalrows != "0")
       <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
       <TR>TD align="left"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRY)</FONT></TD></TR>
       </TABLE></CENTER><BR>
       <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
       <TR><TD VALIGN="top" ><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(CGRYBANNERNAME)</B></FONT>
       </td><td>
       <table BORDER=0 CELLPADDING=4 CELLSPACING=0>
       %WHILE (loop <= totalrows) {
           <tr><td align=right>
           @DTW_TB_GETV(catTable,loop,"1",newcgrfnbr)
           @GET_CHILD_CATS(newcgrfnbr)
           </td></tr>tr><td><br></td></tr>
           @DTW_ADD(loop,"1",loop)
       %}
       </table>
       <BR></TD></TR><TR><TD><BR></TD></TR></TABLE></CENTER>
%ELSE
       @DTW_ASSIGN(NOCATEGORIESINCATEGORY, "YES")
       <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
       <TR><TD align="left"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRY)</FONT></TD></TR>
       </TABLE></CENTER><BR>
       <CENTER><TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>
       <TR><TD VALIGN="top" ><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(CGRYBANNERNAME)</B></FONT>
       <BR></TD></TR><TR><TD><BR></TD></TR>
       </TABLE></CENTER>
%ENDIF
%}
```

D.2 You are here custom page

The code for this custom page is contained in the you_are_here_cat.inc file, which is listed below.

```
%{==========================================================================
== Licensed Materials - Property of IBM                                   ==
==                                                                        ==
== 5697-D24                                                               ==
==                                                                        ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.            ==
==                                                                        ==
== US Government Users Restricted Rights - Use, duplication or disclosure ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                 ==
==========================================================================%}
%define{
hasParent="TRUE"
globalref=""
%}
```

```
%function(dtw_odbc) GET_CAT_PARENT(IN cgrfnbr){
    SELECT cgname, cgrfnbr
    FROM category, cgryrel
    WHERE cgmenbr=$(MerchantRefNum)
       and crmenbr=$(MerchantRefNum)
       and CRCCGNBR=$(cgrfnbr)
       and cgrfnbr=CRPCGNBR
   %REPORT{
       @DTW_ASSIGN(hasParent,"TRUE")
       @DTW_ASSIGN(globalref,V_cgrfnbr)
       setupcategories('$(loop)','$(V_cgname)','$(V_cgrfnbr)')
   %}
   %MESSAGE{
       100:{ @DTW_ASSIGN(hasParent,"FALSE") %} :continue
       default:{@DTW_ASSIGN(hasParent,"FALSE")%} :continue
   %}
%}


%MACRO_FUNCTION DISPLAY_YOU_ARE_HERE(){
<script>
@DTW_ASSIGN(globalref,cgrfnbr)
@DTW_ASSIGN(loop,"0")
%WHILE (loop < "10" && hasParent == "TRUE"){
@GET_CAT_PARENT(globalref)
@DTW_ADD(loop,"1",loop)
%}
var cgmenbr = '$(MerchantRefNum)';
</script>
<tr><td><div align="left">
<font size="$(BodyFontSize)"  face="Arial, Helvetica, sans-serif">
 <img src="/EXT_pages/images/dot_336699.gif" width="7" height="7"> You are here: <a
href="javascript:top.location.reload();">Home</a>
<font size="$(BodyFontSize)" face="Arial, Helvetica, sans-serif">
@DTW_SUBTRACT(loop,"2",loop)
<script>
printyouarehere('$(loop)')
</script>
&#187;$(CGRYBANNERNAME)</font></div></td></tr>
%}
```

D.2.1 you_are_here.js file listning

```
/*****************************************************************************
** Licensed Materials - Property of IBM                                     **
**                                                                          **
** 5697-D24                                                                 **
**                                                                          **
** (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.              **
**                                                                          **
** US Government Users Restricted Rights - Use, duplication or disclosure   **
** restricted by GSA ADP Schedule Contract with IBM Corp.                   **
*****************************************************************************/
var categories = new Array();
var list = "";

function CreateCats(name,ref ) {
this.name=name;
this.ref=ref;
return this;}

function setupcategories(i,name,ref){
```

```
categories[i]=new CreateCats(name,ref );}

function printyouarehere(total){
var index=total;
while( categories[index] && index > -1){
printcategory(index);
index--;
}
document.write(list);}

function printcategory(index){
list += " &#187; ";
list += "<a href=/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=";
list += categories[index].ref+"&cgmenbr="+cgmenbr+">";
list += categories[index].name+"</a>";
}
```

D.3 Shortened product page example, cat_product2.d2w file

```
%{=========================================================================

The sample Templates, HTML and Macros are furnished by IBM as simple
examples to provide an illustration. These examples have not been
thoroughly tested under all conditions.  IBM, therefore, cannot guarantee reliability,
serviceability or function of these programs. All programs contained herein are provided
to you "AS IS".

The sample Templates, HTML and Macros may include the names of individuals,
companies, brands and products in order to illustrate them as completely as
possible.  All of these are names are ficticious and any similarity to the names
and addresses used by actual persons or business enterprises is entirely coincidental.

Licensed Materials - Property of IBM

5648-B47

(C) Copyright IBM Corp.  1998, 1999, 2000.          All Rights Reserved

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


=========================================================================%}

%INCLUDE "/ClassicStoreModel/translation_text.inc"

%define {
    SHOWSQL="NO"
    NC_NOW = "{fn now()}"
         SHOWADDTOCART="NO"
%}

%INCLUDE "/ClassicStoreModel/service_info.inc"

%FUNCTION(dtw_odbc) GET_PRODUCTNAME(){
     SELECT prsdesc
     FROM product
     WHERE prrfnbr=$(prrfnbr) and prmenbr=$(prmenbr) and prpub=1

    %REPORT{
```

```
    %ROW{
           @DTW_assign(PRODUCTNAME, V_prsdesc)
       %}

    %}

    %MESSAGE{
        100:{

           <CENTER>
           <TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4
CELLSPACING=0>

               <TR>
                   <TD ALIGN=$(TitleAlignment) VALIGN="center">
                       <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(ERR_LBL_MESSAGE)</B></FONT>
                   </TD>
               </TR>

               <TR>
                   <TD align="left">
                       <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_PRODNOTAVAILABLE)</B></FONT>
                   </TD>
               </TR>

           </TABLE>

       %}

       default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
GET_PRODUCTNAME()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B> $(DTW_DEFAULT_MESSAGE)</FONT><HR> %}
:continue
    %}
%}


%FUNCTION(dtw_odbc) GET_DISTINCTATTRIBUTES(){
    SELECT count (*) as dattr
    FROM proddstatr
    WHERE pdprnbr=$(prrfnbr) and pdmenbr=$(prmenbr)

    %REPORT{

    %ROW{

           @DTW_ASSIGN(FLAG_ATTRIBUTES, V_dattr)
       %}

    %}
    %MESSAGE{
       100:{No Title%} :continue
       default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
GET_DISTINCTATTRIBUTES()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}
```

```
%function(dtw_odbc) DISPLAY_PRODUCT_IMAGE(){

    SELECT prfull, prvent, prsdesc
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr) and prpub=1

   %REPORT{

    %ROW{

           @DTW_STRIP(@DTW_rSTRIP(V_prfull), CATIMAGE)

           %IF((@DTW_rSUBSTR(CATIMAGE, "1", "1") != "/") && CATIMAGE != NULL)
               @DTW_CONCAT("/yourGallery/merchants/", $(DirectoryName), CATIMAGEPATH)
               @DTW_CONCAT(CATIMAGEPATH, "/", CATIMAGEPATH)
               @DTW_CONCAT(CATIMAGEPATH, CATIMAGE, CATIMAGE)
           %ENDIF

           <TR>
               <TD ALIGN="center">
                   <TABLE BORDER=0  CELLPADDING=4 CELLSPACING=0>
                       %IF (CATIMAGE != NULL)
                           <TR><TD><CENTER><IMG SRC="$(CATIMAGE)" ALT="$(V_prsdesc)"
height=100 width=100></CENTER></TD>
                       %ELSE
                           <!--TR><TD>
                               <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
                               <BR>
                               <CENTER><B>$(MSG_NOIMAGE)</B></CENTER>
                               <BR>
                               </FONT>
                           </TD></TR-->
                       %ENDIF


       %}

       <TD></TD>
   %}
   %MESSAGE{
       100:{ %} :continue
       default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODUCT_IMAGE()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

   %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_INFO(){

    SELECT prldesc1, prnbr, prsdesc
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr)  and prpub=1

   %REPORT{

    %ROW{

           <TD>
               <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(V_prsdesc)</B></FONT>
```

```
                <BR><FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)">[$(LBL_SKU)
$(V_prnbr)]</FONT>
                <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">$(V_prldesc1)</FONT>
                <BR>
        @DTW_ASSIGN(SKUNUMBER, V_prnbr)

    %}

    %}
    %MESSAGE{
        100:{ %} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODUCT_INFO()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}


%function(dtw_odbc) DISPLAY_PRODATTR(){

    SELECT distinct paname, paval
     FROM prodatr
     WHERE pamenbr=$(MerchantRefNum) and paprnbr=$(prrfnbr)

    %REPORT{


        %ROW{

            @DTW_STRIP(V_PAVAL, PAVAL)

                        <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B><I>$(V_paname) : </I></B>$(V_paval)</FONT>
        %}




    %}

    %MESSAGE{
        100:{ %} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODATTR()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B> $(DTW_DEFAULT_MESSAGE)</FONT><HR>
%} :continue

    %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_DUALPRICE(){

    SELECT prnbr, ppprc, ppcur, pppre, ppdeffs, ppdefff,
           NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), 'en_US', 1
) as FormattedCurPrice,
           NC_CurrFormat(NC_CurrConv(ppprc, $(ConvMultiplyFactor), $(ConvDivideFactor)),
'$(ConvCurPrefix)', '$(ConvCurPostfix)', '$(ConvCurGroupingSep)' ,
```

```
'$(ConvCurDecimalSep)' , $(ConvCurRounder), $(ConvCurGroupingSize) ,
$(ConvCurDecimalPlaces), '', 1 ) as FormattedConvCurPrice
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr) and ppprnbr=$(prrfnbr) and
prpub=1
          and (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or (ppdeffs is null and
ppdefff is null))
    ORDER BY pppre DESC

    %REPORT{
         @DTW_ASSIGN(SHOWADDTOCART,"YES")

    @DTW_ASSIGN(ACTUALPRICE, "0")

    <TR BGCOLOR=$(HighLightColor2)>
        <TD ALIGN="center" COLSPAN=2>

    %ROW{

        %IF (ROW_NUM == "1")
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">
            <B>$(LBL_PRODUCTPRICE) : </B>
            </FONT>
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">
            $(V_FormattedCurPrice) $(CurDescription) [$(V_FormattedConvCurPrice)
$(ConvCurDescription)]</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
            @DTW_ASSIGN(ACTUALPRICE, V_ppprc)
        %ENDIF


        %IF (V_pppre == "0" && ACTUALPRICE < V_ppprc)
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)"
COLOR="$(HighlightFontColor2)">
            (<STRIKE>$(V_FormattedCurPrice) $(CurDescription)  [$(V_FormattedConvCurPrice)
$(ConvCurDescription)]</STRIKE>)</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
        %ENDIF


    %}


        </TD>
    </TR>

    <TR><TD><BR></TD></TR>

       <TR>
            <TD align="center">

                <FORM NAME="process"
ACTION="http://$(HOST_NAME)/cgi-bin/ncommerce3/OrderItemUpdate" Method=get>

                <INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>
                <INPUT TYPE=hidden   NAME=product_rn   VALUE=$(prrfnbr)>
                <INPUT TYPE=hidden   NAME=quantity  VALUE=1>
                <INPUT TYPE=hidden   NAME=url
VALUE=https://$(HOST_NAME)/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchantRefNum
)>
```

```
            </TD>
        </TR>

    %IF (SERVICELEVEL == "Basic")
        %INCLUDE "/ClassicStoreModel/cat_prodcmtbox.inc"
    %ELSE
        <TR>
            <TD align="center">
                <BR>
                <INPUT TYPE="hidden" NAME="comment" VALUE="$(LBL_COMMENTS): ">
                <INPUT TYPE=SUBMIT VALUE="@DTW_rUPPERCASE(BUT_ADDTOSHOPCART)">
                </FORM>
            </TD>
        </TR>
    %ENDIF


    %}
    %MESSAGE{
        100:{ No Price%} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODUCT_DUALPRICE()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_SINGLEPRICE(){

    SELECT prnbr, ppprc, ppcur, pppre, ppdeffs, ppdefff,
           NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces), 'en_US', 1
) as FormattedCurPrice
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr) and ppprnbr=$(prrfnbr) and
prpub=1
           and (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or (ppdeffs is null and
ppdefff is null))
    ORDER BY pppre DESC

    %REPORT{

    @DTW_ASSIGN(ACTUALPRICE, "0")
    @DTW_ASSIGN(SHOWADDTOCART,"YES")
    <TR BGCOLOR=$(HighLightColor2)>
        <TD ALIGN="center" COLSPAN=2>

    %ROW{

        %IF (ROW_NUM == "1")
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">
           <B>$(LBL_PRODUCTPRICE) : </B>
           </FONT>
           <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">
           $(V_FormattedCurPrice) $(CurDescription)</FONT>
           <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
           @DTW_ASSIGN(ACTUALPRICE, V_FormattedCurPrice)
       %ENDIF
```

```
        %IF (V_pppre == "0" && ACTUALPRICE < V_FormattedCurPrice)
            <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)"
COLOR="$(HighlightFontColor2)">
            (<STRIKE>$(V_FormattedCurPrice) $(CurDescription)</STRIKE>)</FONT>
            <!--<BR>$(V_ppdeffs) to $(V_ppdefff)-->
        %ENDIF

    %}


        </TD>
    </TR>

       <TR>
            <TD align="center">

                <FORM NAME="process"
ACTION="http://$(HOST_NAME)/cgi-bin/ncommerce3/OrderItemUpdate" Method=get>

                <INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>
                <INPUT TYPE=hidden   NAME=product_rn   VALUE=$(prrfnbr)>
                <INPUT TYPE=hidden   NAME=quantity  VALUE=1>
                <INPUT TYPE=hidden   NAME=url
VALUE=https://$(HOST_NAME)/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchantRefNum
)>

            </TD>
        </TR>

    %IF (SERVICELEVEL == "Basic")
        %INCLUDE "/ClassicStoreModel/cat_prodcmtbox.inc"
    %ELSE
        <TR>
            <TD align="center">
                <BR>
                <INPUT TYPE="hidden" NAME="comment" VALUE="$(LBL_COMMENTS): ">
                <INPUT TYPE=SUBMIT VALUE="@DTW_rUPPERCASE(BUT_ADDTOSHOPCART)">
                </FORM>
            </TD>
        </TR>
    %ENDIF

    %}
    %MESSAGE{
        100:{ No Price%} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODUCT_SINGLEPRICE()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_DUALPRICE_RANGE(){

    SELECT  ppprc, prnbr,
            NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces),
'$(CurLocale)', 1 ) as FormattedCurPrice,
            NC_CurrFormat(NC_CurrConv(ppprc, $(ConvMultiplyFactor), $(ConvDivideFactor)),
'$(ConvCurPrefix)', '$(ConvCurPostfix)', '$(ConvCurGroupingSep)' ,
```

```
'$(ConvCurDecimalSep)' , $(ConvCurRounder), $(ConvCurGroupingSize) ,
$(ConvCurDecimalPlaces), '$(CurLocale)', 1 ) as FormattedConvCurPrice
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum) and prprfnbr=$(prrfnbr) and ppprnbr=prrfnbr and
prpub=1
            and (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or (ppdeffs is null and
ppdefff is null))
    ORDER BY prnbr ASC, pppre DESC

    %REPORT{

        @DTW_ASSIGN(SHOWADDTOCART,"YES")
        %ROW{


            %IF (ROW_NUM == "1")
                @DTW_ASSIGN(STARTPRICE, V_ppprc)
                @DTW_ASSIGN(STARTFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(STARTFORMATTEDCONVPRICE, V_FormattedConvCurPrice)
                @DTW_ASSIGN(STARTPRECEDENCE, V_pppre)
                @DTW_ASSIGN(STARTPRNBR, V_prnbr)

                @DTW_ASSIGN(ENDPRICE, V_ppprc)
                @DTW_ASSIGN(ENDFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(ENDFORMATTEDCONVPRICE, V_FormattedConvCurPrice)
                @DTW_ASSIGN(ENDPRECEDENCE, V_pppre)
                @DTW_ASSIGN(ENDPRNBR, V_prnbr)

                @DTW_ASSIGN(SALEFLAG, "NONE")
                @DTW_ASSIGN(SALEPRNBR, V_prnbr)
                @DTW_ASSIGN(SALEPRICE, V_ppprc)
            %ENDIF

            %IF (SALEFLAG == "NONE" && SALEPRNBR == V_prnbr && V_pppre == "0" && SALEPRICE
< V_ppprc)
                @DTW_ASSIGN(SALEFLAG, "ONSALE")
                @DTW_ASSIGN(SALEPRICE, V_ppprc)
            %ELIF (SALEFLAG == "NONE" && SALEPRNBR != V_prnbr && V_pppre > "0")
                @DTW_ASSIGN(SALEPRICE, V_ppprc)
                @DTW_ASSIGN(SALEPRNBR, V_prnbr)
            %ENDIF

            %IF ((STARTPRNBR == V_prnbr && STARTPRECEDENCE < V_pppre) || (STARTPRNBR !=
V_prnbr && STARTPRICE > V_ppprc))

                @DTW_ASSIGN(STARTPRICE, V_ppprc)
                @DTW_ASSIGN(STARTFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(STARTFORMATTEDCONVPRICE, V_FormattedConvCurPrice)
                @DTW_ASSIGN(STARTPRECEDENCE, V_pppre)
                @DTW_ASSIGN(STARTPRNBR, V_prnbr)
            %ENDIF


            %IF ((ENDPRNBR == V_prnbr && ENDPRECEDENCE < V_pppre) || (ENDPRNBR != V_prnbr
&& ENDPRICE < V_ppprc))

                @DTW_ASSIGN(ENDPRICE, V_ppprc)
                @DTW_ASSIGN(ENDFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(ENDFORMATTEDCONVPRICE, V_FormattedConvCurPrice)
                @DTW_ASSIGN(ENDPRECEDENCE, V_pppre)
                @DTW_ASSIGN(ENDPRNBR, V_prnbr)

                $(V_FormattedCurPrice)
```

```
                $(V_FormattedConvCurPrice)

            %ENDIF
        %}

            <TD BGCOLOR=$(HighLightColor2) ALIGN="center">

                <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">
                <B>$(LBL_PRODUCTPRICE):</B>


                %IF (STARTPRICE == ENDPRICE)
                    $(STARTFORMATTEDPRICE) $(CurDescription) [$(STARTFORMATTEDCONVPRICE)]
                %ELSE
                    $(STARTFORMATTEDPRICE) $(CurDescription) [$(STARTFORMATTEDCONVPRICE)]
$(TRANSLATIONPROB_TXT_TO) $(ENDFORMATTEDPRICE) $(CurDescription)
[$(ENDFORMATTEDCONVPRICE)]
                %ENDIF

                </FONT>

                %IF (SALEFLAG == "ONSALE")
                    <BR><FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">$(MSG_SALEPRICE)</FONT>
                %ENDIF

            </TD>

        %IF (STARTPRICE != ENDPRICE)
                <TD BGCOLOR=$(HighLightColor2) ALIGN=center VALIGN=top>
                    <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSizeSmall)">$(MSG_ITEMSDIFFPRICES)
                    <BR><A HREF=# onClick=ShowPriceList()>$(LINK_PRICEDETAILS)</A>
                    </FONT>
                </TD>

        %ENDIF


    %}
    %MESSAGE{
        100:{ %} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODUCT_DUALPRICE_RANGE()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_SINGLEPRICE_RANGE(){

    SELECT  prrfnbr, ppprc, pppre, prnbr,
            NC_CurrFormat(ppprc, '$(CurPrefix)', '$(CurPostfix)', '$(CurGroupingSep)' ,
'$(CurDecimalSep)' , $(CurRounder), $(CurGroupingSize) , $(CurDecimalPlaces),
'$(CurLocale)', 1 ) as FormattedCurPrice
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum) and prprfnbr=$(prrfnbr) and ppprnbr=prrfnbr and
prpub=1
            and (($(NC_NOW) >= ppdeffs and $(NC_NOW) <= ppdefff) or (ppdeffs is null and
ppdefff is null))
```

```
    ORDER BY prnbr ASC, pppre DESC

    %REPORT{
        @DTW_ASSIGN(SHOWADDTOCART,"YES")
        %ROW{

            %IF (ROW_NUM == "1")
                @DTW_ASSIGN(STARTPRICE, V_ppprc)
                @DTW_ASSIGN(STARTFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(STARTPRECEDENCE, V_pppre)
                @DTW_ASSIGN(STARTPRNBR, V_prnbr)

                @DTW_ASSIGN(ENDPRICE, V_ppprc)
                @DTW_ASSIGN(ENDFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(ENDPRECEDENCE, V_pppre)
                @DTW_ASSIGN(ENDPRNBR, V_prnbr)

                @DTW_ASSIGN(SALEFLAG, "NONE")
                @DTW_ASSIGN(SALEPRNBR, V_prnbr)
                @DTW_ASSIGN(SALEPRICE, V_ppprc)
            %ENDIF

            %IF (SALEFLAG == "NONE" && SALEPRNBR == V_prnbr && V_pppre == "0" && SALEPRICE
< V_ppprc)
                @DTW_ASSIGN(SALEFLAG, "ONSALE")
                @DTW_ASSIGN(SALEPRICE, V_ppprc)
            %ELIF (SALEFLAG == "NONE" && SALEPRNBR != V_prnbr && V_pppre > "0")
                @DTW_ASSIGN(SALEPRICE, V_ppprc)
                @DTW_ASSIGN(SALEPRNBR, V_prnbr)
            %ENDIF

            %IF ((STARTPRNBR == V_prnbr && STARTPRECEDENCE < V_pppre) || (STARTPRNBR !=
V_prnbr && STARTPRICE > V_ppprc))
                @DTW_ASSIGN(STARTPRICE, V_ppprc)
                @DTW_ASSIGN(STARTFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(STARTPRECEDENCE, V_pppre)
                @DTW_ASSIGN(STARTPRNBR, V_prnbr)
            %ENDIF


            %IF ((ENDPRNBR == V_prnbr && ENDPRECEDENCE < V_pppre) || (ENDPRNBR != V_prnbr
&& ENDPRICE < V_ppprc))
                @DTW_ASSIGN(ENDPRICE, V_ppprc)
                @DTW_ASSIGN(ENDFORMATTEDPRICE, V_FormattedCurPrice)
                @DTW_ASSIGN(ENDPRECEDENCE, V_pppre)
                @DTW_ASSIGN(ENDPRNBR, V_prnbr)
            %ENDIF
        %}

            <TD BGCOLOR=$(HighLightColor2) ALIGN="center">

                <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">
                <B>$(LBL_PRODUCTPRICE):</B>

                %IF (STARTPRICE == ENDPRICE)
                    $(STARTFORMATTEDPRICE) $(CurDescription)
                %ELSE
                    $(STARTFORMATTEDPRICE) $(CurDescription) $(TRANSLATIONPROB_TXT_TO)
$(ENDFORMATTEDPRICE) $(CurDescription)
                %ENDIF

                </FONT>
```

```
            %IF (SALEFLAG == "ONSALE")
                <BR><FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor2)">$(MSG_SALEPRICE)</FONT>
            %ENDIF


        %IF (STARTPRICE != ENDPRICE)

            <TD BGCOLOR=$(HighLightColor2) ALIGN=center VALIGN=top>
                <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSizeSmall)">$(MSG_ITEMSDIFFPRICES)
                <BR><A HREF=# onClick=ShowPriceList()>$(LINK_PRICEDETAILS)</A>
                </FONT>

        %ENDIF
    %}
    %MESSAGE{
        100:{ %} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_PRODUCT_SINGLEPRICE_RANGE()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}



%function(dtw_odbc) DISPLAY_ITEMS_DROPDOWN(){

    SELECT DISTINCT paname, paval, pdseqnbr
     FROM prodatr, product, proddstatr
     WHERE pamenbr=$(MerchantRefNum) and paval != 'NULL' and prprfnbr=$(prrfnbr) and
paprnbr=prrfnbr and prpub=1
           and pdmenbr=$(MerchantRefNum) and pdname=paname and pdprnbr=$(prrfnbr)
    ORDER BY pdseqnbr, paname, paval

    %REPORT{

        <FORM ACTION="http://$(HOST_NAME)/cgi-bin/ncommerce3/OrderItemUpdate" Method=get>

        <INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>
        <INPUT TYPE=hidden   NAME=quantity  VALUE=1>
        <INPUT TYPE=hidden   NAME=product_rn   VALUE=$(prrfnbr)>
        <INPUT TYPE=hiddenNAME="prpub"VALUE="active">
        <INPUT TYPE=hidden   NAME=url
VALUE=https://$(HOST_NAME)/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchantRefNum
)>

        <TR>
            <TD ALIGN="Left">

            <TABLE border=0>
            <TR  BGCOLOR=$(HighLightColor1)>
                <TD ALIGN=center>
                <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)">
                $(TXT_INTSR_ITEMSELECT)<BR><BR>
                </FONT>

                @DTW_ASSIGN(TEMP_DDATTR, "")

        %ROW{
```

```
            %IF (V_paname != TEMP_DDATTR && TEMP_DDATTR == "")
                <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor1)"><B>$(V_paname)</b></FONT>
                <SELECT NAME="$(V_paname)">
                <OPTION>-- $(DROPDOWN_SELECT) --</OPTION>
            %ELIF (V_paname != TEMP_DDATTR)
                </SELECT>
                <BR><FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSize)"
COLOR="$(HighlightFontColor1)"><B>$(V_paname)</B></FONT>
                <SELECT NAME="$(V_paname)">
                <OPTION>-- $(DROPDOWN_SELECT) --</OPTION>
            %ENDIF


            <OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>


            @DTW_ASSIGN(TEMP_DDATTR, V_paname)


            </FONT>

        %}

        </SELECT>

        </TD>



    %}

    %MESSAGE{
        100:{%} :continue
        default:{<BR><HR><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>$(ERR_MSG_GENERAL)</B><BR><BR><B>$(ERR_LBL_FUNCTIONNAME)</B>
DISPLAY_ITEMS_DROPDOWN()<BR><B>$(ERR_LBL_ERRORMESSAGE)</B>
$(DTW_DEFAULT_MESSAGE)</FONT><HR> %} :continue

    %}
%}


%MESSAGE{
    4001: {
    <TABLE WIDTH=100% CELLPADDING=0 CELLSPACING=0 BORDER=0>

        <TR>
          <TD ALIGN="left" VALIGN="center">
            <FONT FACE="helvetica"
COLOR=$(TitleTxtCol)><H3>$(ERR_LBL_MESSAGE)</H3></FONT>
          </TD>
        </TR>

        <TR>
            <TD align="left" COLSPAN=3>
                <FONT SIZE=3><B>$(ERR_MSG_INVALIDINPUT)</B></FONT>
            </TD>
        </TR>
```

```
    </TABLE>
    %} :exit
%}

%{================================================%}
%{ HTML Report Section
%{================================================%}

%HTML_REPORT{

<HTML>

@GET_STORE_SERVICELEVEL()
@GET_PRODUCTNAME()
@GET_DISTINCTATTRIBUTES()

<HEAD>
    <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1999 01:01:01 GMT">

    <SCRIPT LANGUAGE="javascript">
        function ShowPriceList() {
        window.open("/cgi-bin/ncommerce3/ExecMacro/" + $(DirectoryName) +
"/cat_prodpricelist.d2w/report?product_rn=" +
$(prrfnbr),"pricelist$(prrfnbr)","toolbar=no,location=no,directories=no,status=no,menuba
r=no,resizable=yes,width=300,height=500,prompt=no,scrollbars=1,setTimeOut=no");
        }
    </SCRIPT>

</HEAD>

<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)" TEXT="$(TextCol)"
LINK="$(LinkCol)" VLINK="$(VLinkCol)" ALINK="$(ALinkCol)">

@DTW_ADD(MerchantRefNum, "0", "2" , SecurityCheck)
@DTW_ADD(prrfnbr, "0", "2" , SecurityCheck)
@DTW_ADD(prmenbr, "0", "2" , SecurityCheck)

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <TR>
        <TD ALIGN=$(TitleAlignment) VALIGN="center">
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(TXT_TITLE_CATALOG)</B></FONT>
        </TD>
    </TR>
</TABLE>
</CENTER>

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

<TR>
    <TD align="left">
        <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)">$(TXT_INSTR_CATALOGCGRY)</FONT>
    </TD>
</TR>

</TABLE>
</CENTER>

<BR>
```

```
<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <TR>
        <TD ALIGN=$(TitleAlignment) VALIGN="center">
            <FONT FACE=$(TitleFontFace) COLOR=$(TitleFontColor)
SIZE=$(TitleFontSize)><B>$(PRODUCTNAME)</B></FONT>
        </TD>
    </TR>



    %IF (REFERENCEPAGE != "CATALOG" && REFERENCEPAGE != "SEARCH")

        <TR><TD><BR></TD></TR>
        <TR>
            <TD align="left">
                <FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSize)"><B>($(TXT_TITLE_PRODUCT))</B></FONT>
            </TD>
        </TR>
    %ENDIF

</TABLE>
</CENTER>

<CENTER>
<TABLE BORDER=0 WIDTH=$(TableWidth) ALIGN=$(TableAlignment) CELLPADDING=4 CELLSPACING=0>

    <!--A
HREF="http://$(HOST_NAME)/cgi-bin/ncommerce3/InterestItemAdd?product_rn=$(prrfnbr)&merch
ant_rn=$(MerchantRefNum)&WISHLISTSHOPPER_RN=$(SESSION_RN)&url=http://$(HOST_NAME)/cgi-bi
n/ncommerce3/InterestItemDisplay">
    <FONT FACE="$(BodyFontFace)" SIZE="$(BodyFontSizeSmall)">Add to my wishlist</FONT>
    </A-->

    @DISPLAY_PRODUCT_IMAGE()
    @DISPLAY_PRODUCT_INFO()
    @DISPLAY_PRODATTR()
    </TD></TR>
    </TABLE></TD></TR>

    %IF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor == NULL || ConvDivideFactor ==
NULL))

        @DISPLAY_PRODUCT_SINGLEPRICE()

    %ELIF (FLAG_ATTRIBUTES == "0" && (ConvMultiplyFactor != NULL && ConvDivideFactor !=
NULL))

        @DISPLAY_PRODUCT_DUALPRICE()

    %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor == NULL || ConvDivideFactor ==
NULL))
        @DISPLAY_ITEMS_DROPDOWN()
        @DISPLAY_PRODUCT_SINGLEPRICE_RANGE()


    %ELIF (FLAG_ATTRIBUTES > "0" && (ConvMultiplyFactor != NULL && ConvDivideFactor !=
NULL))
        @DISPLAY_ITEMS_DROPDOWN()
        @DISPLAY_PRODUCT_DUALPRICE_RANGE()
```

```
    %ENDIF

    %IF (SHOWADDTOCART == "YES")
        <p>
        <INPUT TYPE="hidden" NAME="comment" VALUE="$(LBL_COMMENTS): ">
        <INPUT TYPE=SUBMIT VALUE="@DTW_rUPPERCASE(BUT_ADDTOSHOPCART)">
        </TD>
        </TR>
        </TABLE>
    %ENDIF


    <TR><TD ALIGN="center"><FONT FACE="$(BodyFontFace)"
SIZE="$(BodyFontSizeSmall)">$(MSG_REMOVEFROMSHOPCART)</FONT></TD></TR>

</TABLE>


</BODY>
</HTML>

%}
```

D.4 The “You are here” custom product page and file you_are_here.inc

```
%{==========================================================================
== Licensed Materials - Property of IBM                                   ==
==                                                                        ==
== 5697-D24                                                               ==
==                                                                        ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.            ==
==                                                                        ==
== US Government Users Restricted Rights - Use, duplication or disclosure ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                 ==
==========================================================================%}
%define{
hasParent="TRUE"
globalref=""
%}


%function(dtw_odbc) GET_PROD_CAT(IN prrfnbr){
    SELECT cgname, cgrfnbr
    FROM category, cgprrel
    WHERE cgmenbr=$(MerchantRefNum)
        and cpmenbr=$(MerchantRefNum)
        and cpprnbr=$(prrfnbr)
        and cgrfnbr=cpcgnbr
    %REPORT{
        @DTW_ASSIGN(hasParent,"TRUE")
        @DTW_ASSIGN(globalref,V_cgrfnbr)
        setupcategories(0,'$(V_cgname)','$(V_cgrfnbr)')
    %}
    %MESSAGE{
        100:{ @DTW_ASSIGN(hasParent,"FALSE") %} :continue
        default:{@DTW_ASSIGN(hasParent,"FALSE")%}:continue
    %}
```

```
%}

%function(dtw_odbc) GET_CAT_PARENT(IN cgrfnbr){

    SELECT cgname, cgrfnbr
    FROM category, cgryrel
    WHERE cgmenbr=$(MerchantRefNum)
       and crmenbr=$(MerchantRefNum)
       and CRCCGNBR=$(cgrfnbr)
       and cgrfnbr=CRPCGNBR

   %REPORT{
       @DTW_ASSIGN(hasParent,"TRUE")
       @DTW_ASSIGN(globalref,V_cgrfnbr)
       setupcategories('$(loop)','$(V_cgname)','$(V_cgrfnbr)')
   %}
   %MESSAGE{
       100:{ @DTW_ASSIGN(hasParent,"FALSE") %} :continue
       default:{@DTW_ASSIGN(hasParent,"FALSE")%} :continue

   %}
%}


%MACRO_FUNCTION DISPLAY_YOU_ARE_HERE(){

<script>
@GET_PROD_CAT(prrfnbr)
@DTW_ASSIGN(loop,"1")
%WHILE (loop < "10" && hasParent == "TRUE"){
@GET_CAT_PARENT(globalref)
@DTW_ADD(loop,"1",loop)
%}
var cgmenbr = '$(MerchantRefNum)';
</script>

<tr><td><div align="left">
<font size="$(BodyFontSize)"  face="Arial, Helvetica, sans-serif"> <img
src="/EXT_pages/images/dot_336699.gif" width="7" height="7"> You are here: <a
href="javascript:top.location.reload();">Home</a>
<font size="$(BodyFontSize)" face="Arial, Helvetica, sans-serif">
@DTW_SUBTRACT(loop,"2",loop)

<script>
printyouarehere('$(loop)')
</script>
&#187;$(PRODUCTNAME)
</font></div></td></tr>
%}
```

D.5 Full macros XML parameter

```
<?xml version="1.0" encoding="iso-8859-1"?>
<macros>
 <path>macro/common/ClassicStoreModel</path>
   <task>
     <refno>16</refno>
     <file>logon.d2w</file>
  </task>
   <task>
     <refno>1013</refno>
```

```
    <file>logon.d2w</file>
</task>
 <task>
    <refno>7</refno>
    <file>ord_details.d2w</file>
</task>
 <task>
    <refno>8</refno>
    <file>ord_details.d2w</file>
</task>
 <task>
    <refno>9</refno>
    <file>ord_details.d2w</file>
</task>
 <task>
    <refno>1006</refno>
    <file>ord_details.d2w</file>
</task>
 <task>
    <refno>1016</refno>
    <file>ord_details.d2w</file>
</task>
 <task>
    <refno>3</refno>
    <file>ord_billto.d2w</file>
</task>
 <task>
    <refno>1015</refno>
    <file>err_addr.d2w</file>
</task>
 <task>
    <refno>5</refno>
    <file>ord_ok.d2w</file>
</task>
 <task>
    <refno>4</refno>
    <file>ord_ok.d2w</file>
</task>
 <task>
    <refno>1028</refno>
    <file>err_dopay.d2w</file>
</task>
 <task>
    <refno>10</refno>
    <file>reg_new.d2w</file>
</task>
 <task>
    <refno>11</refno>
    <file>reg_update.d2w</file>
</task>
 <task>
    <refno>1002</refno>
    <file>err_reg.d2w</file>
</task>
 <task>
    <refno>1010</refno>
    <file>err_reg.d2w</file>
</task>
 <task>
    <refno>1014</refno>
    <file>err_reg.d2w</file>
</task>
 <task>
```

```
    <refno>9030</refno>
    <file>logon_pwdreset.d2w</file>
</task>
 <task>
    <refno>9034</refno>
    <file>logon_pwdreset.d2w</file>
</task>
 <task>
    <refno>9037</refno>
    <file>logon_pwdreset.d2w</file>
</task>
 <task>
    <refno>9038</refno>
    <file>logon_pwdchange.d2w</file>
</task>
 <task>
    <refno>9032</refno>
    <file>logon_pwdchange.d2w</file>
</task>
 <task>
    <refno>9031</refno>
    <file>logon_pwdchange.d2w</file>
</task>
 <task>
    <refno>9035</refno>
    <file>logon_pwdchange.d2w</file>
</task>
 <task>
    <refno>1005</refno>
    <file>ord_none.d2w</file>
</task>
 <task>
    <refno>50</refno>
    <file>err_paywakeup.d2w</file>
</task>
 <task>
    <refno>1050</refno>
    <file>err_paywakeup.d2w</file>
</task>
 <task>
    <refno>13</refno>
    <file>ord_updaddr.d2w</file>
</task>
 <task>
    <refno>1001</refno>
    <file>cat_items.d2w</file>
</task>
 <task>
    <refno>1008</refno>
    <file>cat_attributes.d2w</file>
</task>
 <task>
    <refno>9043</refno>
    <file>cat_category.d2w</file>
</task>
 <task>
    <refno>9042</refno>
    <file>cat_product.d2w</file>
</task>
 <task>
    <refno>6</refno>
    <file>wish_list.d2w</file>
</task>
```

```
<name>ClassicStoreModel</name>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_catalog</name>
    <generator>cat_catalog</generator>
 </macro>
  <macro>
    <baseDir>categoryBase</baseDir>
    <name>cat_category</name>
    <generator>cat_category</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_category</name>
    <generator>cat_category</generator>
 </macro>
  <macro>
    <baseDir>productBase</baseDir>
    <name>cat_product</name>
    <generator>cat_product</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_product</name>
    <generator>cat_product</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_items</name>
    <generator>cat_items</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_prodpricelist</name>
    <generator>cat_prodpricelist</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>cat_attributes</name>
    <generator>cat_attributes</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_details</name>
    <generator>ord_details</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_billto</name>
    <generator>ord_billto</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_shipto</name>
    <generator>ord_shipto</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_update</name>
    <generator>ord_update</generator>
 </macro>
  <macro>
    <baseDir>macroBase</baseDir>
```

```
    <name>ord_pay</name>
    <generator>ord_pay</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>err_dopay</name>
    <generator>err_dopay</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>err_paywakeup</name>
    <generator>err_paywakeup</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_ok</name>
    <generator>ord_ok</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_none</name>
    <generator>ord_none</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>ord_status</name>
    <generator>ord_status</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>reg_new</name>
    <generator>reg_new</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>reg_update</name>
    <generator>reg_update</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>err_reg</name>
    <generator>err_reg</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>err_addr</name>
    <generator>err_addr</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>logon</name>
    <generator>logon</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>logon_idforgot</name>
    <generator>logon_idforgot</generator>
</macro>
 <macro>
    <baseDir>macroBase</baseDir>
    <name>logon_pwdchallenge</name>
    <generator>logon_pwdchallenge</generator>
</macro>
```

```
   <macro>
     <baseDir>macroBase</baseDir>
     <name>logon_pwdchange</name>
     <generator>logon_pwdchange</generator>
   </macro>
   <macro>
     <baseDir>macroBase</baseDir>
     <name>logon_pwdreset</name>
     <generator>logon_pwdreset</generator>
   </macro>
   <macro>
     <baseDir>macroBase</baseDir>
     <name>cservice</name>
     <generator>cservice</generator>
   </macro>
   <macro>
     <baseDir>macroBase</baseDir>
     <name>br_turnoff</name>
     <generator>br_turnoff</generator>
   </macro>
   <macro>
     <baseDir>macroBase</baseDir>
     <name>search_display</name>
     <generator>search_display</generator>
   </macro>
</macros>
```

D.6 CheckForChanges.js JavaScript

```
/*****************************************************************************
** Licensed Materials - Property of IBM                                     **
**                                                                          **
** 5697-D24                                                                 **
**                                                                          **
** (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.              **
**                                                                          **
** US Government Users Restricted Rights - Use, duplication or disclosure   **
** restricted by GSA ADP Schedule Contract with IBM Corp.                   **
*****************************************************************************/

// arrays for changed Elements
var newElements = new Array();

function CreateElement( orig, newvalue ) {
this.orig = orig
this.newvalue=newvalue;
return this;
}

function UpdateElement( orig, newvalue, index ) {
newElements[index].orig=orig;
newElements[index].newvalue=newvalue;
return this;
}

function getarrayindex_Elements( orig ) {
var newindex=newElements.length;
var b = true;
var index=0;
while( newElements[index] && b){
if (newElements[index].orig == orig)
```

```
    {b=false;newindex=index;break;}
index++;
}
return newindex;
}

function setXMLChanges(orig_page, chosen_page, orig_base , chosen_base, orig_gen,
chosen_gen)
{
var i=getarrayindex_Elements( orig_page );
if (i==newElements.length)
newElements[i]=new CreateElement(orig_page,chosen_page);
else
UpdateElement(orig_page,chosen_page,i);

i=getarrayindex_Elements( orig_base );
if (i==newElements.length)
newElements[i]=new CreateElement(orig_base,chosen_base);
else
UpdateElement(orig_base,chosen_base,i);

i=getarrayindex_Elements( orig_gen );
if (i==newElements.length)
newElements[i]=new CreateElement(orig_gen,chosen_gen);
else
UpdateElement(orig_gen,chosen_gen,i);
}

//the following fns put the changes into the final XML.

function nospaces(tempstring)
{
var tempstring2="";
for ( j=0; j < tempstring.length; j++)
    {if (tempstring.charAt(j) != " ")
        tempstring2 += tempstring.charAt(j);
    }
return tempstring2;
}

function changeXML(xmlstring) {
nlcode = "10";
var tempstring="";
var tempstring2="";
var newxmlstring="";
var index = 0;
var b = false;

for ( i=0; i < xmlstring.length; i++)
{
if (xmlstring.charCodeAt(i) == nlcode )
    {
    // go around all elements and check for a hit. If yes then change it.
    index = 0;
    b = false;
    tempstring2 = nospaces(tempstring);
    while( newElements[index] && !b ) {
        if ( newElements[index].orig == tempstring2 ){
            tempstring = newElements[index].newvalue;
            b = true;
            break;
        } // if
        index++;
```

```
    } //while
    newxmlstring += tempstring + xmlstring.charAt(i);
    tempstring="";
    }
else
    tempstring += xmlstring.charAt(i);
}
return newxmlstring;
}
```

D.7 DisplayPagesServlet.java servlet

```
/*****************************************************************************
** Licensed Materials - Property of IBM                                     **
**                                                                          **
** 5697-D24                                                                 **
**                                                                          **
** (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.              **
**                                                                          **
** US Government Users Restricted Rights - Use, duplication or disclosure   **
** restricted by GSA ADP Schedule Contract with IBM Corp.                   **
*****************************************************************************/

/*********************************************************************
* DisplayPagesServlet
*********************************************************************/

// These imports are for the XML parsing
import org.w3c.dom.*;
import com.ibm.xml.parsers.*;
import com.ibm.xml.parser.*;

import java.io.*;
import java.beans.Beans;

// These are for the servlet methods
import javax.servlet.*;
import javax.servlet.http.*;

//We import our custom created Bean
import macrosBean;

/*********************************************************************
* DisplayPagesServlet - This servlet creates an instance of a Bean
* (macroBean) from an XML file. And then redirects to a specified
* jsp file for display of this beans attributes
* Usage : /servlet/DisplayPagesServlet?
*         xmlfile = <filename>
*         &jspfile= <jsp file>
*   initParam: xmlpath = <directory path>
*  History
*  =======
*  Version  Date      Who  Comment
*  -------  --------  ---  -------
*   1.0   00/04/01   AH   Initial
*********************************************************************/

public class DisplayPagesServlet extends HttpServlet
{

   public void doGet(HttpServletRequest req, HttpServletResponse res)
```

```
      throws ServletException, IOException
  {
        performTask(req, res);
    }

   public void doPost(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
  {
        performTask(req, res);
    }

   public void performTask(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException
  {

     // Get parameters both init and from request.
    String xmlValue = getServletConfig().getInitParameter("xmlpath") +
req.getParameter("xmlfile");
    String jspValue = req.getParameter("jspfile");

      macrosBean mBean;

    // create an instance of the macrosbean
      try
     {
         mBean = (macrosBean) Beans.instantiate(this.getClass().getClassLoader(),
"macrosBean");
      }
      catch (Exception ex)
      {
         throw new ServletException("Can't create BEAN of class macrosBean: "
                                    + ex.getMessage());
      }

      // Call the XML parse function
   parseXML(xmlValue,mBean);

      // To send the Bean to a JSP file for content formatting and display
      // 1) Set the Bean as an attribute in the current request object
      ((com.sun.server.http.HttpServiceRequest) req).setAttribute("mBean", mBean);

      // 2) Use callPage to invoke the JSP file and pass the current request object
      ((com.sun.server.http.HttpServiceResponse) res).callPage(jspValue, req);

   } // end of performTask method



   public void parseXML(String xmlpath, macrosBean mBean)
      throws ServletException, IOException
  {

    try{
    // declare a DOMparser on our xml file.
    DOMParser parser = new DOMParser();
    parser.parse(xmlpath);
    // Then we retreive the document root of the file.
    Document doc = null;
    doc = parser.getDocument();
    Element root = (Element)doc.getDocumentElement();

    // Having retrieved the root we use that to retreive the other elements by tag name.
    // which we then use as parameters in assign calls to the macroBean
```

```
    mBean.settitle(((Element)
root.getElementsByTagName("title").item(0)).getAttribute("value"));
    mBean.setorigpage(((Element)
root.getElementsByTagName("origpage").item(0)).getAttribute("value"));
    mBean.setorigbase(((Element)
root.getElementsByTagName("origbase").item(0)).getAttribute("value"));

    //Here we get the list of elements containing the newtask tag
    NodeList elements = root.getElementsByTagName("newtask");

    //Then we loop around this list to obtain the other tag values
    for (int i = 0; i < elements.getLength(); i++) {
    Element orderItem = (Element)elements.item(i);
    mBean.setfile(((Element)elements.item(i)).getAttribute("file"),i);
    mBean.setbase(((Element)elements.item(i)).getAttribute("base"),i);
    mBean.setimage(((Element)elements.item(i)).getAttribute("image"),i);
    mBean.setdesc(((Element)elements.item(i)).getAttribute("desc"),i);
    }

    }
    catch(FileNotFoundException e1){
    e1.printStackTrace();
    }
    catch(Exception e2){
    e2.printStackTrace();
    }

} //end of parseXML method

} /* end of class DisplayPagesServlet */
```

D.8 MacrosBean.java class

```
/*****************************************************************************
** Licensed Materials - Property of IBM                                     **
**                                                                          **
** 5697-D24                                                                 **
**                                                                          **
** (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.              **
**                                                                          **
** US Government Users Restricted Rights - Use, duplication or disclosure   **
** restricted by GSA ADP Schedule Contract with IBM Corp.                   **
*****************************************************************************/

import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;

public class macrosBean extends HttpServlet {
private String title = "";
private String origpage = "";
private String origbase = "";
private static final int numpages = 3;
private String[] file = new String[numpages];
private String[] base = new String[numpages];
private String[] image = new String[numpages];
private String[] desc = new String[numpages];

public void service(HttpServletRequest req, HttpServletResponse res)
throws IOException
{
```

```
}
public void settitle(String value) {
this.title = value;
}
public String gettitle() {
return title;
}

public void setorigpage(String value) {
this.origpage = value;
}
public String getorigpage() {
return origpage;
}
public void setorigbase(String value) {
this.origbase = value;
}
public String getorigbase() {
return origbase;
}

public void setfile(String value, int i) {
this.file[i] = value;
}
public String getfile(int i) {
if (file != null)
        return file[i];
else
    return "default";
}

public void setbase(String value, int i) {
this.base[i] = value;
}
public String getbase(int i) {
if (base != null)
        return base[i];
else
    return "default";
}

public void setimage(String value, int i) {
this.image[i] = value;
}
public String getimage(int i) {
if (image != null)
        return image[i];
else
    return "default";
}

public void setdesc(String value, int i) {
this.desc[i] = value;
}
public String getdesc(int i) {
if (desc != null)
        return desc[i];
else
    return "default";
}
}
```

D.9 DisplayPages.jsp page

```
<html><head>
<title>DisplayPages</title>
<!-- Get the Bean using the BEAN tag   -->

<!--Licensed Materials - Property of IBM -->
<!--5697-D24-->
<!--(C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.-->
<!--US Government Users Restricted Rights - Use, duplication or disclosure-->
<!--restricted by GSA ADP Schedule Contract with IBM Corp.-->

<bean name="mBean" type="macrosBean" introspect="no" create="no" scope="request"></bean>

<script>

function Initialize(){
   top.pageAlreadyLoaded = true;
   setselected(document.forms[0]);
}

//getting the variables from the bean.
<% out.println("var orig_page='<file>" + mBean.getorigpage() +"</file>';" );%>
<% out.println("var orig_basetemp='" + mBean.getorigbase() +"';");%>

var orig_base="<name>" + orig_basetemp +"</name>";
var orig_gen="<generator>" + orig_basetemp +"</generator>";

//theOptions variable holds the different page choices for the user
var theOptions = new Array();
var list = "";

function CreateOptions(file, baseDir ) {
this.file=file;
this.baseDir=baseDir;
return this;
}

//This function prints out each choice on the panel and sets it's values in variable
theOptions.
function setuppage(filevalue,basevalue,j,image,desc)
{
var listbase="<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>";
var listbase2 = "<td><p></td><td><input type=radio name=choice value=" +j + "
onclick=storesettings(this.value)></td>";
var listimage = "<img src=/EXT_pages/images/" + image+" width=130 height=150>"
list += listbase2 + listbase + listimage +"</td></tr><tr><td><font face=Arial, Helvetica,
sans-serif>"+ desc +"</font></td></tr></table></td>";
theOptions[j]=new CreateOptions(filevalue,basevalue);
}

//prints out all the choices on the panel
function printoptions(){
document.write(list);
}

//When a merchant chooses a page the setXMLChanges fn is called
//to create or update that choice in the newElements array
function storesettings(index){
top.setXMLChanges(orig_page,'<file>'+theOptions[index].file+'</file>',orig_base,'<name>'
+theOptions[index].baseDir+'</name>',orig_gen,'<generator>'+theOptions[index].baseDir+'<
/generator>')
```

```
}

// These functions display the previously selected option if the merchant is returning
// to this page or sets the default selected one.

function setselected(form)
{
var chosen=0;
for (var i in form.choice){
    if (was_selected(theOptions[i].file)){
        chosen=i;
        break;}
}
form.choice[chosen].checked = true
}

function was_selected(svalue){
var b = false;
var index=0;
while( top.newElements[index] && !b){
temp = '<file>' + svalue + '</file>';
if (top.newElements[index].orig == orig_page && top.newElements[index].newvalue == temp)
    {b=true;break;}
index++;
}
return b;
}


</script>
</head>
<body onLoad="Initialize()">
<p><B><font size=5>
<!-- Print the title value from the macrosBean  -->
<insert bean=mBean property=title default="Please choose from below."
></insert></font></B><p>

<form>
<table border="0" CELLPADDING=2 CELLSPACING=2><tr>

<script>
<REPEAT INDEX="i">

<% out.println("setuppage('" + mBean.getfile(i)+"','"+
mBean.getbase(i)+"',"+i+",'"+mBean.getimage(i)+"','"+mBean.getdesc(i)+"')" );%>

</REPEAT>
printoptions();
</script>

<td><font face="Arial, Helvetica, sans-serif"></font></td></tr>
</table>
</form>
</body>
</html>
```

D.10 HTML panel pages and file Displaycats./

```
<html><head>
<title>DisplayPages</title>
<script>
```

```
<!--Licensed Materials - Property of IBM -->
<!--5697-D24-->
<!--(C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.-->
<!--US Government Users Restricted Rights - Use, duplication or disclosure-->
<!--restricted by GSA ADP Schedule Contract with IBM Corp.-->


function Initialize(){
   top.pageAlreadyLoaded = true;
   setuppage();
   setselected(document.forms[0]);
}

var orig_page='<file>cat_category.d2w</file>';
var orig_base="<name>cat_category</name>";
var orig_gen="<generator>cat_category</generator>";

//theOptions variable holds the different page choices for the user
var theOptions = new Array();
var list = "";

function CreateOptions(file, baseDir ) {
this.file=file;
this.baseDir=baseDir;
return this;
}

//This function prints out each choice on the panel and sets it's values in variable
theOptions.
function setuppage()
{
theOptions[0]=new CreateOptions('cat_category.d2w','cat_category');
theOptions[1]=new CreateOptions('cat_category1.d2w','cat_category1');
theOptions[2]=new CreateOptions('cat_category2.d2w','cat_category2');
}

//When a merchant chooses a page the setXMLChanges fn is called
//to create or update that choice in the newElements array
function storesettings(index){
top.setXMLChanges(orig_page,'<file>'+theOptions[index].file+'</file>',orig_base,'<name>'
+theOptions[index].baseDir+'</name>',orig_gen,'<generator>'+theOptions[index].baseDir+'<
/generator>')
}

// These functions display the previously selected option if the merchant is returning
// to this page or sets the default selected one.

function setselected(form)
{
var chosen=0;
for (var i in form.choice){
    if (was_selected(theOptions[i].file)){
        chosen=i;
        break;}
}
form.choice[chosen].checked = true
}

function was_selected(svalue){
var b = false;
var index=0;
while( top.newElements[index] && !b){
```

```
temp = '<file>' + svalue + '</file>';
if (top.newElements[index].orig == orig_page && top.newElements[index].newvalue == temp)
    {b=true;break;}
index++;
}
return b;
}


</script>
</head>
<body onLoad="Initialize()">
<p><B><font size=5>
Please choose from the category pages below:</font></B><p>

<form>
<table border="0" CELLPADDING=2 CELLSPACING=2><tr>

<td><p></td><td><input type=radio name=choice value="0"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>
<img src=/EXT_pages/images/cat1.gif width=130 height=150>
</td></tr><tr><td><font face=Arial, Helvetica, sans-serif>Default
page</font></td></tr></table></td>

<td><p></td><td><input type=radio name=choice value="1"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>
<img src=/EXT_pages/images/cat2.gif width=130 height=150>
</td></tr><tr><td><font face=Arial, Helvetica, sans-serif>More
Categories</font></td></tr></table></td>

<td><p></td><td><input type=radio name=choice value="2"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>
<img src=/EXT_pages/images/cat3.gif width=130 height=150>
</td></tr><tr><td><font face=Arial, Helvetica, sans-serif>You_Are_Here
Info</font></td></tr></table></td>

<td><font face="Arial, Helvetica, sans-serif"></font></td></tr>
</table>
</form>
</body>
</html>
```

D.11 Displayprods.html

```
<html><head>
<title>DisplayPages</title>
<script>

<!--Licensed Materials - Property of IBM -->
<!--5697-D24-->
<!--(C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.-->
<!--US Government Users Restricted Rights - Use, duplication or disclosure-->
<!--restricted by GSA ADP Schedule Contract with IBM Corp.-->

function Initialize(){
   top.pageAlreadyLoaded = true;
   setuppage();
   setselected(document.forms[0]);
```

```
}

var orig_page='<file>cat_product.d2w</file>';
var orig_base="<name>cat_product</name>";
var orig_gen="<generator>cat_product</generator>";

//theOptions variable holds the different page choices for the user
var theOptions = new Array();
var list = "";

function CreateOptions(file, baseDir ) {
this.file=file;
this.baseDir=baseDir;
return this;
}

//This function prints out each choice on the panel and sets it's values in variable
theOptions.
function setuppage()
{
theOptions[0]=new CreateOptions('cat_product.d2w','cat_product');
theOptions[1]=new CreateOptions('cat_product2.d2w','cat_product2');
theOptions[2]=new CreateOptions('cat_product3.d2w','cat_product3');
}

//When a merchant chooses a page the setXMLChanges fn is called
//to create or update that choice in the newElements array
function storesettings(index){
top.setXMLChanges(orig_page,'<file>'+theOptions[index].file+'</file>',orig_base,'<name>'
+theOptions[index].baseDir+'</name>',orig_gen,'<generator>'+theOptions[index].baseDir+'<
/generator>')
}

// These functions display the previously selected option if the merchant is returning
// to this page or sets the default selected one.

function setselected(form)
{
var chosen=0;
for (var i in form.choice){
    if (was_selected(theOptions[i].file)){
        chosen=i;
        break;}
}
form.choice[chosen].checked = true
}

function was_selected(svalue){
var b = false;
var index=0;
while( top.newElements[index] && !b){
temp = '<file>' + svalue + '</file>';
if (top.newElements[index].orig == orig_page && top.newElements[index].newvalue == temp)
    {b=true;break;}
index++;
}
return b;
}


</script>
</head>
<body onLoad="Initialize()">
```

```
<p><B><font size=5>
Please choose from the product pages below:</font></B><p>

<form>
<table border="0" CELLPADDING=2 CELLSPACING=2><tr>

<td><p></td><td><input type=radio name=choice value="0"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>
<img src=/EXT_pages/images/prod1.gif width=130 height=150>
</td></tr><tr><td><font face=Arial, Helvetica, sans-serif>Default
page</font></td></tr></table></td>

<td><p></td><td><input type=radio name=choice value="1"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>
<img src=/EXT_pages/images/prod2.gif width=130 height=150>
</td></tr><tr><td><font face=Arial, Helvetica, sans-serif>Shortened
Page</font></td></tr></table></td>

<td><p></td><td><input type=radio name=choice value="2"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2><tr><td bgcolor=black>
<img src=/EXT_pages/images/prod3.gif width=130 height=150>
</td></tr><tr><td><font face=Arial, Helvetica, sans-serif>You_are_here
Info</font></td></tr></table></td>


<td><font face="Arial, Helvetica, sans-serif"></font></td></tr>
</table>
</form>
</body>
</html>
```

D.12 Advanced.xml includes

Here are the new panel statements when using the html files above. A dummy parameter is set as the servlet MerchantAdmin will add some parameters by default so we need to add a dummy first to make it work.

```
<panel name="storeCreatorPanelCategory"
url="/EXT_pages/html/displaycats.html?test=nothing" />

<panel name="storeCreatorPanelProduct"
url="/EXT_pages/html/displayprods.html?test=nothing" />
```

D.13 Configurations for samples and Directory structure

A base directory was created /appldata/. And subdirectories were as follows:

HTML - holds html & jsp files.
JavaScript - for .js files.
macro - for macros and include files.

XML - for xml files.
source - source Java code.

D.13.1 HTML, JavaScript, jsp and macro configuration

The following pass statements were added to the httpd.conf file.

```
Pass    /EXT_NCTools/* /appldata/*
Pass    /EXT_pages/* /appldata/*
```

The following added to the net.data db2www.ini file.

```
MACRO_PATH /appldata/macro; ...
INCLUDE_PATH /appldata/macro;...
```

D.13.2 Websphere configuration

The following jar files are required in the classpath when compiling the Java code.

For macrosBean:

```
/usr/jdk_base/lib/classes.zip
/usr/WebSphere/AppServer/lib/jsdk.jar
```

For DisplayPagesServlet:

```
/appldata/source/
/usr/jdk_base/lib/classes.zip
/usr/WebSphere/AppServer/lib/jsdk.jar;
/usr/WebSphere/AppServer/lib/jst.jar
/usr/lpp/NetCommerce3/Tools/lib/xml4j.jar
```

Copy the class files to directory:

```
/usr/WebSphere/AppServer/servlets/
```

To run DisplayPagesServlet you need to create an initial parameter for the servlet. We do this through the websphere configuration interface which is at the following url.

```
http://<hostname>:9527/
```

Choose **Servlets** > **Configuration** from the left hand menu. Add a servlet entry for our DisplayPagesServlet with initial parameter xmlpath which equals /appldata/xml/. See Figure 163.

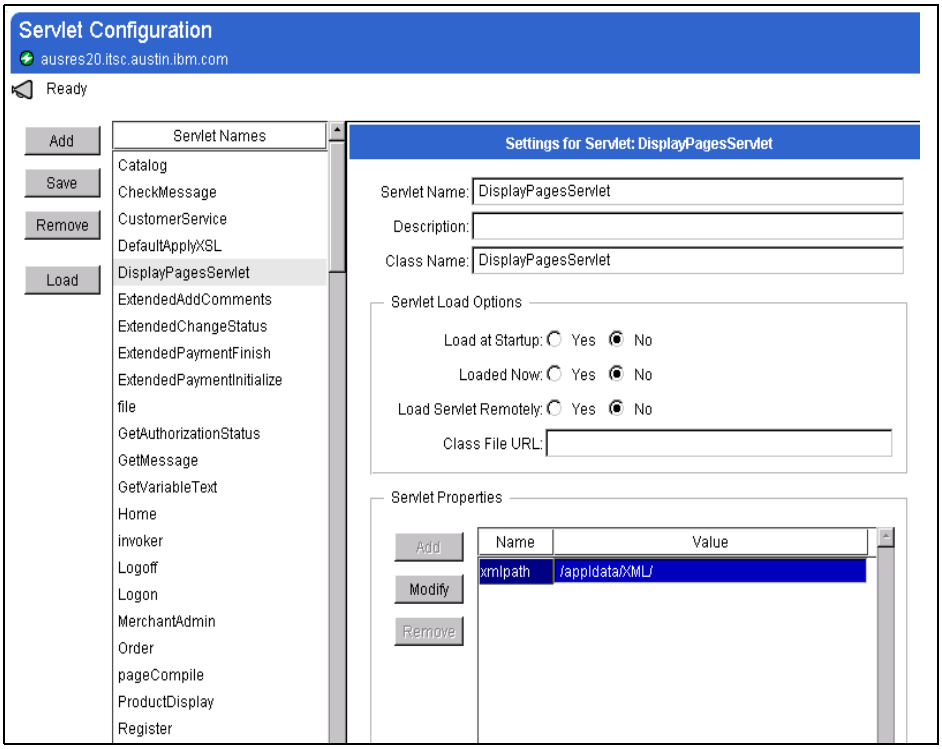


Figure 163. Configuration panel for DisplayPagesServlet

Appendix E. Shipping by weight source code

This appendix contains source code listing for programs used in Section 9.1, “Shipping by product weight” on page 161.

E.1 Macro file shipbywdata.d2w

```
%{=============================================================================
== Licensed Materials - Property of IBM                                     ==
==                                                                          ==
== 5697-D24                                                                 ==
==                                                                          ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved. ==
==                                                                          ==
== US Government Users Restricted Rights - Use, duplication or disclosure =
== restricted by GSA ADP Schedule Contract with IBM Corp.
==============================================================================%}

%DEFINE
{
  DATABASE="demomall"
  LOGIN="db2inst1"
  PASSWORD="chuy5"
  DTW_HTML_TABLE="YES"
  prmenbr=""
%}

%FUNCTION (DTW_ODBC) updateDb()
{
  update PRODUCT set PRWGHT = $(weight) , prwmeas = '$(unit)', PRPSNBR= (
select PSRFNBR from prspcode where PSMENBR = $(prmenbr) and PSSPMTHD =
'W3')
   where PRRFNBR = $(product_ref)
 %}


%FUNCTION (DTW_ODBC) getproduct()
{
   select PRRFNBR,PRMENBR,PRNBR,PRSDESC,PRWGHT,PRWMEAS from PRODUCT where
PRMENBR=$(merfnbr)
   %REPORT
   {
     %ROW
     {
     <OPTION VALUE="$(V_PRRFNBR)"> $(V_PRMENBR)-$(V_prnbr)-$(V_prsdesc)
     </OPTION>
     %}
   %}
%}



%FUNCTION(DTW_ODBC) getProductInfo() {
  SELECT PRRFNBR, PRMENBR, PRNBR, PRSDESC, PRWGHT, PRWMEAS
  FROM PRODUCT WHERE PRRFNBR=$(product_ref)
%REPORT{
%ROW{
   <p>
   Product number and description:
```

```
   <p>
   $(V_PRNBR) - $(V_PRSDESC)<P>
   <p>
   <p>
   <p>
   Product weight - Please update values
   <p>
   <INPUT TYPE="text" NAME="weight" VALUE="$(V_PRWGHT)"> -
   <SELECT NAME="unit">
   <OPTION VALUE="$(V_PRWMEAS)"> $(V_PRWMEAS)
   <OPTION VALUE="Kg"> Kg
   <OPTION VALUE="Lbs"> Lbs
   </OPTION>
   </SELECT>
   <INPUT TYPE="hidden" NAME="product_ref" VALUE="$(V_prrfnbr)">
%}
%}
%}

%FUNCTION (DTW_ODBC) getproduct2() {
   select PRRFNBR,PRMENBR,PRNBR,PRSDESC,PRWGHT,PRWMEAS from PRODUCT where
PRRFNBR=$(product_ref)
   %REPORT {
     %ROW {
     Product Number and description
     <br>
     <INPUT TYPE="text" NAME="tmp1" VALUE="$(V_PRNBR) - $(V_PRSDESC)">
     <p>
     <p>
     Product weight
     <br>
     <INPUT TYPE="text" NAME="tmp3" VALUE="$(V_PRWGHT) - $(V_PRWMEAS)">
     %}
   %}
%}


%HTML_REPORT
{
 <HTML>
  <HEAD> <TITLE> weight based shipping</TITLE>
  </HEAD>

  <BODY>
   <FORM METHOD="GET" ACTION="INPUT">
   @DTW_ASSIGN(prmenbr,merfnbr)
   <P>

   Updating Products for Merchant: $(merfnbr)
   <INPUT TYPE="HIDDEN" NAME="prmenbr" VALUE="$(prmenbr)">
   <P>

   Select product to be updated: <SELECT NAME="product_ref">
   @getproduct()
   </SELECT>
   <P>
   <P>
   <P>
   <INPUT TYPE="submit" VALUE="Get product">
   <P>
   </FORM>
  </BODY>
 </HTML>
```

```
%}


%HTML_INPUT
{
 <HTML>
  <HEAD>
  </HEAD>
   <BODY>
   <p>
   <BR>
   <FORM ACTION="PROCESS">

   You selected the following product: $(product_ref)<P>
   @getProductInfo()
   <p>
   <p>
   <BR>
   <INPUT TYPE="SUBMIT" VALUE="update">
   <INPUT TYPE="HIDDEN" NAME="prmenbr" VALUE="$(prmenbr)">
   </FORM>
   <p>
   </BODY>
 </HTML>
%}

%HTML(PROCESS)
{
 <HTML>
  <HEAD>
  </HEAD>
   <BODY>
   <FORM >
       @updateDb()
       You have succesfully updated the product
       <p>
       <p>
       @getproduct2()
       <p>
       <p>
   </FORM>
   </BODY>
 </HTML>
%}
```

E.2 Macro file shipbyweight.d2w

```
%{=================================================================
== Licensed Materials - Property of IBM                          ==
==                                                               ==
== 5697-D24                                                      ==
==                                                               ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved. =
==                                                               ==
== US Government Users Restricted Rights - Use, duplication or disclosure==
== restricted by GSA ADP Schedule Contract with IBM Corp. ==
=================================================================%}

%DEFINE
```

```
{
  DATABASE="demomall"
  LOGIN="db2inst1"
  PASSWORD="chuy5"
  DTW_HTML_TABLE="YES"
  pradr=""
  prcontry=""
  prstartamt=""
  prendamt=""
  prcharge=""
%}

%FUNCTION (DTW_ODBC) updateDb()
{
 insert into SHIPPING values( (select case when max(sprfnbr) is null then 1
else max(sprfnbr) + 1 end from shipping),$(spmenbr),(select  MMRFNBR from
mshipmode  where MMMENBR = $(spmenbr)),(select PSRFNBR from PRSPCODE where
PSMENBR = $(spmenbr) and PSSPMTHD = 'W3'
),'$(pradr)','$(prcontry)',$(prstartamt),$(prendamt),$(prcharge),null,null
,null,null,null,null)
%}


%FUNCTION (DTW_ODBC) selectcontry()
{
   select CONTNAME
 from COUNTCODE
   %REPORT
   {
     %ROW
     {
   <OPTION VALUE="$(V_CONTNAME)"> $(V_CONTNAME)
   </OPTION>
     %}
   %}
%}

%FUNCTION (DTW_ODBC) selectstate()
{
   select STANAME
 from STATCODE  where STACNTRY = '$(prcontry)'
   %REPORT
   {
     %ROW
     {
   <OPTION VALUE="$(V_STANAME)"> $(V_STANAME)
   </OPTION>
     %}
   %}
%}


%HTML_REPORT
{
 <HTML>
  <HEAD> <TITLE> product Catalog shipping weight editor </TITLE>
  </HEAD>

   @DTW_ASSIGN(spmenbr,merfnbr)

  <BODY>
   <FORM METHOD="GET" ACTION="PROCESS">
   <P>
```

```
   Maintaining Shipping table for merchant: $(spmenbr)
   <P>
   <P>
   <P>
   <P>
   Select contry for shipping cost  : <SELECT NAME="prcontry">
   @selectcontry()
   </SELECT>
   <P>
   <P>
   <P>
   <BR>
   <INPUT TYPE="HIDDEN" VALUE="$(spmenbr)" NAME="spmenbr">
   <INPUT TYPE="SUBMIT" VALUE="proceed">

  </FORM>
  </BODY>
 </HTML>
%}

%HTML(PROCESS)
{
 <HTML>
  <HEAD> <TITLE> product Catalog shipping weight editor </TITLE>
  </HEAD>
   <BODY>
   <FORM METHOD="GET" ACTION="PROCESS2">
   <p>
   <p>
   Contry selected for update is: $(prcontry)
   <p>
   Select state for shipping cost: <SELECT NAME="pradr">
   @selectstate()
   </SELECT>
   <p>
   Enter start weight range in Kg : <INPUT TYPE="TEXT" size="5" VALUE="1"
NAME="prstartamt">
   <p>
   Enter end weight range in Kg   : <INPUT TYPE="TEXT" size="5" VALUE="1"
NAME="prendamt">
   <p>
   Enter price for shipping       : <INPUT TYPE="TEXT" size="6" VALUE="1"
NAME="prcharge">
   <p>

   <BR>

   <INPUT TYPE="HIDDEN" VALUE="$(spmenbr)" NAME="spmenbr">
   <INPUT TYPE="HIDDEN" VALUE="$(prcontry)" NAME="prcontry">
   <p>
   <INPUT TYPE="SUBMIT" VALUE="update">

    </FORM>
   </BODY>
 </HTML>
%}


%HTML(PROCESS2)
{
 <HTML>
  <HEAD>
  </HEAD>
```

```
  <BODY>
   <p>
   Maintaining Shipping table for merchant = $(spmenbr)
   <p>
   Contry selected for update is: $(prcontry)
   <p>
   <BR>
   @updateDb()
   You have updated the shipping information.
  </BODY>
 </HTML>
%}
```

E.3 Macro file shipbywenable.d2w

```
%{=========================================================================
== Licensed Materials - Property of IBM                                  ==
==                                                                       ==
== 5697-D24                                                              ==
==                                                                       ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved. ==
==                                                                       ==
== US Government Users Restricted Rights - Use, duplication or disclosure =
== restricted by GSA ADP Schedule Contract with IBM Corp. ==
=========================================================================%}

%DEFINE
{
  DATABASE="demomall"
  LOGIN="db2inst1"
  PASSWORD="chuy5"
  DTW_HTML_TABLE="YES"
  NEWMTHD=""
%}

%FUNCTION (DTW_ODBC) updateDb()
{
 update prspcode set PSSPMTHD = '$(NEWMTHD)'
 where  psmenbr = $(spmenbr) and PSSPDESC ='Weight_Based'
%}


%FUNCTION (DTW_ODBC) selectship()
{
select PSSPMTHD  from prspcode where psmenbr = $(spmenbr) and PSSPDESC =
'Weight_Based'
   %REPORT
   {
     %ROW
     {
   <OPTION VALUE="$(V_PSSPMTHD)"> $(V_PSSPMTHD)
   </OPTION>
     %}
   %}
%}
```

```
%HTML_REPORT
{
 <HTML>
  <HEAD> <TITLE> weight  </TITLE>
  </HEAD>

   @DTW_ASSIGN(spmenbr,merfnbr)

  <BODY>
   <FORM METHOD="GET" ACTION="PROCESS">
   <P>
   Maintaining Shipping table  merchant = $(spmenbr)
   <P>
   Current shipping method is: $(V_PSSPMTHD)
   @selectship()
   <P>
   <P>
   enter walue Q3 or W3 :

   <SELECT NAME="NEWMTHD">
   <OPTION VALUE="Q3"> Q3
   <OPTION VALUE="W3"> W3
   </OPTION>
   </SELECT>
   <p>

   <BR>
   <INPUT TYPE="HIDDEN" VALUE="$(spmenbr)" NAME="spmenbr">
   <INPUT TYPE="SUBMIT" VALUE="update">

  </FORM>
  </BODY>
 </HTML>
%}


%HTML(PROCESS)
{
 <HTML>
  <HEAD>
  </HEAD>
   <BODY>
   Maintaining Shipping table for merchant = $(spmenbr)
   @updateDb()
   <p>
   <p>
   <BR>
       You have updated the shipping information.
  </BODY>
 </HTML>
%}
```

E.4 C++ function GetOrdByWt.cpp

This section list the C++ code for the Overridable function GetOrdByWt. This function implement a new shipping method where the shipping cost calculation is based on weight rather than by units.

```
//
//============================================================================================
//
// Licensed Materials - Property of IBM
//
// (c)  Copyright  IBM Corp. 1995, 1997, 1998, 1999, 2000.      All Rights Reserved
//
// US Government Users Restricted Rights - Use, duplication or
// disclosure restricted by GSA ADP Schedule Contract with IBM Corp
//
// GetOrdByWt.cpp - Overridable function for task GET_ORD_SH_TOT
//
// Implicit input parameters:
// ORDER_REF_NUM  : The reference number of the order
//
//
//
//============================================================================================
#include "nc_core.pch"
#include "objects/objects.pch"

#if defined (WIN32)
#define __DLL_NCS_API__ __declspec(dllexport)
#elif defined(AIX)
#define __DLL_NCS_API__
#endif

//============================================================================================
#ifdef __TRACE_NCAPIS__
typedef TraceYes Trace;
#else
typedef TraceNo  Trace;
#endif
static Trace trace("NC_APIS ("__FILE__")");
//============================================================================================

class __DLL_NCS_API__ GetOrdByWt : public NC_OverridableFunction
{
  static const ClassName _STR_ThisClass;

public:
  GetOrdByWt() {
    Trace::Tracer T(_STR_CONSTRUCTOR, _STR_ThisClass);
  }

  virtual ~GetOrdByWt() {
    Trace::Tracer T(_STR_DESTRUCTOR, _STR_ThisClass);
  }

  void  operator delete( void* p ) { ::delete p; }

public:
   virtual bool Process(const HttpRequest& Req, HttpResponse& Res, NC_Environment& Env)
{
       Trace::Tracer T("ProcessGiftMsg", _STR_ThisClass);
```

```
       static const StringWithOwnership _PARAM_NAME_PRODUCT_SHIPPING
("PRODUCT_SHIPPING");

       //
*****************************************************************************
       // the parameter ORDER_REF_NUM is defined by the Net.Commerce API
       //
       const String& OrderRefNum = *(const String *)Env.Seek("ORDER_REF_NUM");
       long ordernumber = 0;
       long merchantnumber = 0;
       OrderRefNum.getVal(ordernumber);
       _MerchantRefNum.getVal(merchantnumber);

       trace << indent << "GetOrdByWt: order_rn = " << OrderRefNum
           << ", merchant_rn = " << merchantnumber << endl;

       //
*****************************************************************************
       // Find out if we got a Q3(default) method in our basket.
       //
       String stmt;
       Row SqlRow;
       long Error;
       DataBase* DB = DataBaseManager::GetCurrentDataBase();

       stmt.Clean();
       stmt << "select prrfnbr, prpsnbr "
            << "from product, shipto "
            << "where prrfnbr=stprnbr and "
            << "stornbr=" << OrderRefNum << " and "
            << "ststat='P' and "
            << "prpsnbr is null";

       SQL getShipMthdSql(*DB, stmt);
       if (getShipMthdSql.getSQLrc() != ERR_DB_NO_ERROR) {
         getShipMthdSql.ReportError();
         error << indent << "GetOrdByWt: failed quering for shipping." << endl;
         return false;
       }

       //
*****************************************************************************
       // If a Q3 method is found, call default GetOrdShTot OF
       if ((Error = getShipMthdSql.getNextRow(SqlRow)) != ERR_DB_NO_DATA) {
          debug << indent << "GetOrdByWt: calling default OF GetOrdShTot " << endl;

           static const StringWithOwnership _TASK_GET_ORD_SH_TOT     ("GET_ORD_SH_TOT");
       const ErrorMsg_Cmd* Err = NC_OverridableFunctionManager::Call(Req,
                                 Res,
                                 Env,
                                 _TASK_GET_ORD_SH_TOT,
                                 merchantnumber
                                 );

           if (Err == &_ERR_CMD_ERR_HANDLED)
          throw Err;

          return true;
       }

       trace << indent << "GetOrdByWt: calculating new shipping." << endl;
```

```
        //
*****************************************************************************
        // Calculate shipping based on weight
        //
        // find weight of all items in shopping basket
        stmt.Clean();
        stmt << "select sum(stquant*prwght) as totalwght "
             << "from product,shipto,prspcode "
             << "where prrfnbr=stprnbr and "
             << "stornbr=" << OrderRefNum << " and "
             << "ststat='P' and "
             << "prpsnbr = psrfnbr and "
             << "psspmthd = 'W3'";

        SQL getWghtTotSql(*DB, stmt);
        if (getWghtTotSql.getSQLrc() != ERR_DB_NO_ERROR) {
          getWghtTotSql.ReportError();
          error << indent << "GetOrdByWt: failed quering total weight." << endl;
          return false;
        }
        getWghtTotSql.getNextRow(SqlRow);
        double wghttot = 0.0;
        SqlRow.getCol("totalwght").Trim().getVal(wghttot);
        trace << indent << "GetOrdByWt: Total weight = " << wghttot << endl;

        //
        // Fetch shipping address
        // If now rows found, use 'Other'
        stmt.Clean();
        stmt << "select spstramt, spendamt, spchrge "
             << "from shipping, shaddr "
             << "where spcntry = sacntry and "
             << "spaddrjr = sastate and "
             << "sarfnbr in (select stsanbr from shipto where stornbr=" << OrderRefNum <<
") and "
             << "spmenbr=" << merchantnumber << " and "
             << "(spstramt <= " << wghttot << " and " << wghttot << " < spendamt)";
        SQL getRangeSql(*DB, stmt);
        if (getRangeSql.getSQLrc() != ERR_DB_NO_ERROR) {
          getRangeSql.ReportError();
          error << indent << "GetOrdByWt: failed quering range." << endl;
          return false;
        }
        if ((Error = getRangeSql.getNextRow(SqlRow)) == ERR_DB_NO_DATA) {
           // Use 'Other' - must be defined
           stmt.Clean();
           stmt << "select spstramt, spendamt, spchrge "
                << "from shipping, shaddr "
                << "where spcntry = 'Other' and "
                << "sarfnbr in (select stsanbr from shipto where stornbr=" << OrderRefNum
<< ") and "
                << "spmenbr=" << merchantnumber << " and "
                << "(spstramt <= " << wghttot << " and " << wghttot << " < spendamt)";
            SQL getRange2Sql(*DB, stmt);
            if (getRange2Sql.getSQLrc() != ERR_DB_NO_ERROR) {
              getRange2Sql.ReportError();
              error << indent << "GetOrdByWt: failed quering range for 'Other'." << endl;
              return false;
            }
           getRange2Sql.getNextRow(SqlRow);
        }

        // return shipping price to calling command or of
```

```
        String* Shipping = (String*) Env.Seek(_PARAM_NAME_PRODUCT_SHIPPING);
        *Shipping << SqlRow.getCol("spchrge");

        return true;
    }
};


const ClassName GetOrdByWt::_STR_ThisClass("GetOrdByWt");
static bool X = NC_OverridableFunctionManager::GetUniqueInstance().RegisterApi
("IBM ITSO", "WCS SPE", "GetOrdByWt", 1.0, new GetOrdByWt);

static String Copyright = "(c) Copyright IBM Corp. 1995, 1997, 1998, 1999, 2000. All
Rights Reserved";
```

E.5 Makefile for GetOrdByWt.cpp

This makefile constuct the library file libITSO.a from the C++ file GetOrdByWt.cpp, the make file perform all steps to generate the libray file from the C++ source code.

```
##
================================================================================
##
================================================================================
##
## FILE NAME:    makefile
##
## DESCRIPTION:  makefile to build the ITSO lib for Net.Commerce
##
##
================================================================================

GCC              = xlC_r
LINK             = /usr/lpp/xlC/bin/linkxlC_r
LINKSHR          = /usr/ibmcxx/bin/makeC++SharedLib_r
MKLIB            = ar -v -r -u
INCLUDE          = -I/usr/lpp/NetCommerce3/adt/include
CFLAGS           = $(DEBUG) -w -DAIX -qlistopt -qlist -qxref=full -c $(INCLUDE)
GCAPILIBFLAGS    = -+ $(DEBUG) -DAIX -DNLS_ENABLED -DCLNK #-DSYSCTRL_UNIX

LIBS             = -L./ -L/usr/lpp/xlC/lib -lld -L/usr/lpp/NetCommerce3/adt/lib \
          -lC -lnc3_dbc \
                   -lserver_objs \
          -lnc3_messages \
          -lnc3_common \
          -lnc3_containers \

LIBPATH          = .

SONAME           = libITSO.a

OBJS      =GetOrdByWt.o
```

```
all             : $(OBJS) $(SONAME)

#addGiftMsg.o  : addGiftMsg.cpp
#          $(GCC) $(CFLAGS) addGiftMsg.cpp

GetOrdByWt.o  : GetOrdByWt.cpp
           $(GCC) $(CFLAGS) GetOrdByWt.cpp

$(SONAME): $(OBJS)
       $(LINKSHR) -bloadmap:libITSO.loadmap -p0 \
                        -o $(SONAME)  $(OBJS) $(LIBS)

clean           :
       @echo -
       @echo -
       @echo == Cleaning... ================
       @echo -
       rm *.o *.lst
       rm $(LIBPATH)/$(SONAME)
```

E.6 reg_GetOrdByW.db2.sql SQL script

This SQL script will register the function GetOrdByWt in WCS as an oververriable function.

```
******************************
SQL example
******************************



delete from ofs where name ='GetOrdByWt' and dll_name = 'libITSO.a';
-- ****************************************************************************
-- * Register of GetOrdByWt for PROCESS GET_ORD_SH_TOT
-- *
insert into ofs(refnum, dll_name, vendor, product, name, version)
values(
(select max(refnum)+1 from ofs),
'libITSO.a',
'IBM ITSO','WCS SPE',
'GetOrdByWt',1.0
);
```

Appendix F. Payment method code lists

This section contains a listing of the PaymentWizard.xml file from Section 9.2, “Adding off-line payment methods” on page 198.

F.1 PaymentWizard.xml

```
<?xml version="1.0"?>
<!--
  This XML file defines the behaviour of the payment wizard. By modifying this file,
  you can customize in the following ways:
     * add a new payment authorization method
     * configure the payment wizard to use SET
     * add new panels to the payment wizard
     * add/remove credit cards and modify their images
  ____________________________________________________________________________

  The 'finishPage' attribute defines the URL to call when a merchant clicks
  finish button on the last page of the payment wizard. The following shows
  how the URL is constructed the parameters that are passed:

   framework.finishURL + "&XML=<paymentWizard xml data....>"
                       + "&merchant.refno=12345"

  By defining this URL in this XML file, the default finish logic can be easily
  replaced by your own custom logic.

  In addition, the extendedFinishURL can be defined to call your own code after the
  current finish routine has completed.  To do this, simply uncomment the
  extendedFinishURL tag below chich calls the ExtendedPaymentFinish sample  servlet.
  You can modify or replace this servlet.
  ____________________________________________________________________________

  Payment authorization methods are defined using the 'paymentMethod' tag.  This XML
  file defines the payment methods for CyberCash and SET using the IBM Payment
  Server. Note: Net.Commerce only supports a single payment method to be enable/mall.
  By default, CyberCash is enabled and SET is not.  To enable SET, uncomment the
  paymentMethod tag that describes this method. If you uncomment the SET payment
  method make sure you comment the CyberCash payment method and its corresponding
panel.

Additional payment methods may be added to wizard by defining new paymentMethod tags.
  See the example below, that shows how to enable a custom payment method.

  ____________________________________________________________________________-->


<framework rootName="payment"
           resourceBundle="nchs.paymentNLS"
           title="paymentWizardTitle"
           finishConfirmation="finishConfirmation"
           cancelConfirmation="canelConfirmation"
           initializeURL="CTnchs.payment.Initialize"
           finishURL="https://$env.hostname$/servlet/MerchantAdmin?PROCESS=CTnchs.payment.Finish" >

    <!--
```

```
      <extendedInitializeURL>http://localhost/servlet/ExtendedPaymentInitialize</extendedInitializeURL>
      <extendedFinishURL>http://localhost/servlet/ExtendedPaymentFinish</extendedFinishURL>
      -->

  <!--
    Define the panels that will be displayed in the payment
    wizard. Note that the order of these panels defines
    the order they are displayed in the wizard.
    -->

  <panel name = "welcomeTab"
         url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.Welcome"
         hasHelp   = "no" />

  <panel name ="chooseOnlineOfflineTab"
         url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.ChooseOnlineOffline"
         helplink = "CTnchs.Payment.chooseOnlineOffline.Help" />

  <panel name = "offlineInstructionsTab"
         url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.OfflineInstructions"
         helplink = "CTnchs.Payment.offlineInstructions.Help" />

  <panel name = "chooseOnlineMethodsTab"
         url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.ChooseOnlineMethods"
         helplink = "CTnchs.Payment.chooseOnlineMethods.Help" />

  <!--
    This example shows how to add a configuration page for your own
    payment method.  The configPage attribute points to the
    panel that will show your configuration page.  Uncomment this
    example and the corresponding panel tag to see an example.

    The checkBoxName and checkBoxDescription are tags used to extract national
    language strings from a resource bundle to make translation easier.  See
    custom.nchs.payment.NLStrings.properties to change the names of these tags.
-->

  <!--
  <paymentMethod name="PayConfigExample"
                 checkBoxName="customCheckBoxName"
                 checkBoxDescription="customCheckBoxDesc"
                 configPage="payConfigExampleTab" />

  <panel name = "payConfigExampleTab"
         url  = "/NCTools/html/nchs/$env.locale$/samples/payment/PayConfigExample.html"
         helplink = "" />
    -->

  <!-- CyberCash payment method -->
  <paymentMethod name="CYBERCASH"
                 checkBoxName="cyberCashCheckBoxName"
                 checkBoxDescription="cyberCashCheckBoxDesc"
                 configPage="cyberCashConfigTab" />

  <panel name = "cyberCashConfigTab"
         url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.CyberCashConfig"
         helplink = "CTnchs.Payment.cyberCashConfig.Help" />

<panel name = "onlineInstructionsTab"
         url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.OnlineInstructions"
```

```
        helplink = "CTnchs.Payment.onlineInstructions.Help" />

  <panel name = "summaryTab"
        url  = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.Summary"
        hasHelp   = "no"
        hasFinish ="yes" />

  <!--
    Uncomment the following paymentMethod section to enable the payment wizard
    for SET using the IBM Payment Server.  Make sure you comment out the above
    CyberCash payment method and its corresponding configuation panel.
    -->

  <!--
    <paymentMethod name="SET"
                  checkBoxName="SETCheckBoxName"
                  checkBoxDescription="SETCheckBoxDesc"
                  configPage="" />
    -->

  <!-- Define the layout of the frames in the wizard. -->
  <!-- Define the layout of the frames in the wizard. -->
  <frame type="frameset"
        value="ROWS=38,*,55"
        options="BORDER=0">
      <frame    type    = "frame"
                name    = "Banner"
                src     = "/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.Banner"



options = "border=0 frameborder=0 noresize scrolling=no marginwidth=0 marginheight=0" />
      <frame type="frameset" value="COLS=160,5,*" options="BORDER=0">
        <frame    type    = "frame"
                  name    = "TOC"
                  src     = "/NCTools/html/nchs/paymentTableOfContents.html"
                  options = "border=0 frameborder=0 noresize marginwidth=0 marginheight=0" />
        <frame    type    = "frame"
                  name    = "Divider"
                  src     = "/NCTools/html/divider.html"
                  options = "border=0 frameborder=0 noresize marginwidth=0 marginheight=0" />
        <frame    type       = "frame"
                  name       = "CONTENTS"
                  isContents = "yes"
                 options    = "BORDER=0 FRAMEBORDER=0 NORESIZE SCROLLING=NO MARGINWIDTH=15 MARGINHEIGHT=15"
/>
      </frame>
      <frame    type    = "frame"
                src     =
"/servlet/MerchantAdmin?DISPLAY=CTnchs.payment.Navigation&amp;XMLFile=nchs.payment.paymentWizard.xml"
                name    = "NAVIGATION"
                options = "BORDER=0 FRAMEBORDER=0 NORESIZE SCROLLING=NO MARGINWIDTH=0 MARGINHEIGHT=0" />
   </frame>


</framework>
```

F.2 OfflineInstructions.tem tewmplate code

```
Model nchs(paymentNLS)

display()
{
/*
<!--=======================================================
Licensed Materials - Property of IBM

5648-B47

(c)  Copyright  IBM Corp.  1998, 1999.      All Rights Reserved

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
=======================================================-->
<HTML>
<LINK REL="stylesheet" HREF="/NCTools/html/common/pagestyle.css">
<SCRIPT>
  var payment = top.get( "payment" );
  function initializeState()
  {
    document.f1.paymentInstructions1.value = payment.offlineInstructions;
    document.f1.confirmMessage1.value = payment.offlineConfirmation;
  }

  function validateEntries()
  {
    if ( document.f1.paymentInstructions1.value.length > 256 ||
         document.f1.confirmMessage1.value.length > 256 ) {
       alert ("$paymentNLS.offlineMsgTooLong$");
       return false;
    }
    payment.offlineInstructions = document.f1.choseninstr.value;
    payment.offlineConfirmation = document.f1.confirmMessage1.value;
    top.put( "payment", payment );
    return true;
  }
</SCRIPT>
<BODY BGCOLOR="#CCCCCC" ONLOAD="initializeState();">
<FORM NAME="f1">
$paymentNLS.offlineDesc$<BR>
<UL TYPE="DISC">
<input type=radio name=choose
onclick="this.form.choseninstr.value=this.form.paymentInstructions1.value;" CHECKED>
    <B>$paymentNLS.offlineInstructions$</B>
    <BR>
    <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions1" ROWS=2 COLS=35>Call us to a
number 1-800-___-____ to bring your credit card information and order number.</TEXTAREA>
<BR>
<BR>
<input type=radio name=choose
onclick="this.form.choseninstr.value=this.form.paymentInstructions2.value;">
    <B>Pay with check</B>
    <BR>
```

```
    <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions2" ROWS=2 COLS=35 >Send us your
check to the account number xxx-xxxxxxxx of the xxxxxxxxxxxx bank to complete your
order.</TEXTAREA>
<BR>
<BR>
<input type=radio name=choose
onclick="this.form.choseninstr.value=this.form.paymentInstructions3.value;">
    <B>Cash On Delivery (C.O.D.)</B>
    <BR>
    <TEXTAREA WRAP=PHYSICAL NAME="paymentInstructions3" ROWS=2 COLS=35 >Just pay your
order when you recive it in your home. As easy as it!</TEXTAREA>
<input type=hidden name=choseninstr value="">
<BR>
<BR>
<BR>
<B>$paymentNLS.offlineConfirmation$</B>
<BR>
<TEXTAREA WRAP=PHYSICAL NAME="confirmMessage1" ROWS=1 COLS=35>We appreciate a lot your
purchase. For any information about your order, please see the Customer Information
Section.</TEXTAREA>

  </UL>
</FORM>
</BODY>
</HTML>
*/
}
```

Appendix G. Source code for gift message/wrapping

This section contains program listings for the examples used in Section 9.3, “Gift messages/wrapping” on page 205. The information is also available in downloadable format from the ITSO Web page.

G.1 HTML file giftadmin.htm

```
<HTML>
<HEAD>
<TITLE>Manage Gift Message/Wrapping Feature</TITLE>
</HEAD>
<FRAMESET   ROWS="30%,*"    FRAMEBORDER="no"    BORDER=0    BORDERCOLOR="white">
    <FRAME  NAME="topframe"     SRC=
"/msprotect/ncommerce3/ExecMacro/ncadmin/sitemgr/giftadmin.d2w/report"
MARGINWIDTH=0 MARGINHEIGHT=32
           FRAMEBORDER="no"    BORDERCOLOR="white"             SCROLLING="auto">
    <FRAME  NAME="mainframe"    SRC="/butnbars/s_blank.htm"      MARGINWIDTH=0
MARGINHEIGHT=0
           FRAMEBORDER="no"    BORDERCOLOR="white"             SCROLLING="auto">
</FRAMESET>
</HTML>
```

G.2 Macro file giftadmin.d2w

```
%{============================================================================
== Licensed Materials - Property of IBM                                     ==
==                                                                          ==
== 5697-D24                                                                 ==
==                                                                          ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.              ==
==                                                                          ==
== US Government Users Restricted Rights - Use, duplication or disclosure   ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                   ==
============================================================================%}

%include "ncadmin/common.d2w"

%include "ncadmin/adminchk.d2w"

%{===== NOTES: ==============================================================
  1. See 'ncadmin/common.d2w' for definition of variables access_control_query_*.
  ============================================================================%}

%{============================================================================
  This function retrieves all merchants and displays a nice dropdown menu
  ============================================================================%}
%FUNCTION(DTW_ODBC) StoreName() {
    SELECT DISTINCT
    merfnbr, mestname
    FROM
    merchant
    WHERE
        merfnbr in (select mpmenbr from mcspinfo)
```

```
    ORDER BY
    mestname
    %REPORT{
    <b>Store Name</b><BR>
    <SELECT NAME="store_rn">
    %ROW{
        <OPTION VALUE="$(V_MERFNBR)">$(V_MESTNAME)</OPTION>
    %}
    </SELECT> <INPUT TYPE="submit" VALUE="Find Store">

    %}
    %MESSAGE{
    default: { <P><FONT SIZE=+1>No stores could be found!</FONT></BODY></HTML> %} : exit
    %}
%}

%HTML_REPORT{
<!-- ensure that user has permission to use this page -->
@AbortIfAdminAccessDeniedForID(SESSION_ID)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<TITLE>Manage Gift Message/Wrapping Feature</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<H2>Manage Gift Message/Wrapping Feature</H2>
<FORM ACTION="/msprotect/ncommerce3/ExecMacro/ncadmin/sitemgr/giftadmin2.d2w/report"
TARGET="mainframe">
<!-- fetch all the stores -->
@StoreName()

</FORM>
</BODY>
</HTML>
%}
```

G.3 Macro file giftadmin2.d2w

```
%{============================================================================
== Licensed Materials - Property of IBM                                     ==
==                                                                          ==
== 5697-D24                                                                 ==
==                                                                          ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.              ==
==                                                                          ==
== US Government Users Restricted Rights - Use, duplication or disclosure   ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                   ==
==                                                                          ==
== This macro takes the MERFNBR as the input parameter named 'store_rn'     ==
==                                                                          ==
== and retrieves gift message/wrapping informations about the store         ==
============================================================================%}

%include "ncadmin/common.d2w"

%include "ncadmin/adminchk.d2w"

%{===== NOTES: ==============================================================
```

```
  1. See 'ncadmin/common.d2w' for definition of variables access_control_query_*.
  ================================================================================%}

%{================================================================================
  This function retrieves all merchants and displays a nice dropdown menu
  If the merchant do not have an entry in the GIFTMESSAGE table, we will set a flag
  to create a new entry, instead of updating the existing row
  ================================================================================%}
%FUNCTION(DTW_ODBC) GetGiftInformations() {
    SELECT
    gmmenbr, gmallow, gmenabled, gmdesc, merfnbr
    FROM
    giftmessage, merchant
    WHERE
        merfnbr = $(store_rn) AND
        gmmenbr = merfnbr

    %REPORT{
        %ROW{
            @DTW_assign(gmmenbr, V_gmmenbr)
            @DTW_assign(gmallow, V_gmallow)
            %IF(V_gmallow == "0")
                @DTW_assign(gmallowEnabled, "")
                @DTW_assign(gmallowDisabled, "checked")
            %ELSE
                @DTW_assign(gmallowEnabled, "checked")
                @DTW_assign(gmallowDisabled, "")
            %ENDIF
            @DTW_assign(gmenabled, V_gmenabled)
            %IF(V_gmenabled == "1")
                @DTW_assign(enabledInStore, "yes")
            %ELSE
                @DTW_assign(enabledInStore, "no")
            %ENDIF
            @DTW_assign(gmdesc, V_gmdesc)
            @DTW_assign(merfnbr, V_merfnbr)
            @DTW_assign(create_new_entry, "no")
        %}
    %}
    %MESSAGE{
        100: {@DTW_assign(create_new_entry, "yes")
                @DTW_assign(gmallowEnabled, "")
                @DTW_assign(gmallowDisabled, "checked")
                @DTW_assign(enabledInStore, "no")
         %} : continue
    default: {
             <P><FONT SIZE=+1><B>Database Error:</B><P>
             A database error occurred during the database search.
             Please contact the Net.Commerce server administrator.</FONT>
             <P>SQL Code = $(SQL_CODE), SQL State = $(SQL_STATE)</BODY></HTML>
        %} : exit
    %}
%}


%FUNCTION(DTW_ODBC) createNewEntry(IN merfnbr, IN status) {
    INSERT INTO
        giftmessage(gmmenbr, gmallow)
    VALUES($(merfnbr), $(status))
%}


%FUNCTION(DTW_ODBC) updateEntry(IN merfnbr, IN status) {
```

```
    UPDATE
        giftmessage
    SET
        gmallow = $(status)
    WHERE
        gmmenbr = $(merfnbr)
%}


%HTML_REPORT{
<!-- ensure that user has permission to use this page -->
@AbortIfAdminAccessDeniedForID(SESSION_ID)

%if(store_rn != "" && create == "yes")
@createNewEntry(store_rn, gmallow)
@CommitSQL()
%elif(store_rn != "" && create == "no")
@updateEntry(store_rn, gmallow)
@CommitSQL()
%endif

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<TITLE>Manage Gift Message/Wrapping Feature</TITLE>
</HEAD>

<BODY BGCOLOR="#FFFFFF">
<!-- fetch informations about the merchant -->
@GetGiftInformations()

<FORM TARGET="mainframe">
<INPUT TYPE="hidden" NAME="store_rn" VALUE="$(store_rn)">
<INPUT TYPE="hidden" NAME="create" VALUE="$(create_new_entry)">
<b>Gift Message/Wrapping status</b><BR>
<INPUT TYPE="radio" NAME="gmallow" VALUE="1" $(gmallowEnabled)>Enabled<BR>
<INPUT TYPE="radio" NAME="gmallow" VALUE="0" $(gmallowDisabled)>Disabled<BR>
<INPUT TYPE="submit" VALUE="UPDATE STATUS">
</FORM>
<H3>Informations handled by merchant</H3>
Enabled in store: <b>$(enabledInStore)</b><BR>
Information text: <br>
$(gmdesc)
</BODY>
</HTML>
%}
```

G.4 Macro file giftadminstore.d2w

```
%{=============================================================================
== Licensed Materials - Property of IBM                                      ==
==                                                                           ==
== 5697-D24                                                                  ==
==                                                                           ==
== (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.               ==
==                                                                           ==
== US Government Users Restricted Rights - Use, duplication or disclosure    ==
== restricted by GSA ADP Schedule Contract with IBM Corp.                    ==
```

```
==                                                                            ==
== This macro takes the MERFNBR as the input parameter named 'merfnbr'       ==
==                                                                            ==
== and retrieves gift message/wrapping informations for this store           ==
================================================================================%}

%include "ncadmin/common.d2w"

%include "ncadmin/adminchk.d2w"

%{===== NOTES: ==========================================================================
  1. See 'ncadmin/common.d2w' for definition of variables access_control_query_*.
  ==========================================================================================%}

%FUNCTION(DTW_ODBC) GetGiftInformations() {
    SELECT
    gmmenbr, gmallow, gmenabled, gmdesc, mestname
    FROM
    giftmessage, merchant
    WHERE
        merfnbr = $(merfnbr) AND
        gmmenbr = merfnbr AND
        gmallow = 1

    %REPORT{
        %ROW{
            @DTW_assign(ENABLED, "yes")
            @DTW_assign(gmenabled, V_gmenabled)
            %IF(V_gmenabled == "1")
                @DTW_assign(enabledInStore, "checked")
                @DTW_assign(disabledInStore, "")
            %ELSE
                @DTW_assign(enabledInStore, "")
                @DTW_assign(disabledInStore, "checked")
            %ENDIF
            @DTW_assign(gmdesc, V_gmdesc)
            @DTW_assign(LongStoreName, V_mestname)
        %}
    %}

    %MESSAGE{
        100 : { @DTW_assign(ENABLED, "no")
         %} : continue
    default: {
             <P><FONT SIZE=+1><B>Database Error:</B><P>
             A database error occurred during the database search.
             Please contact the Net.Commerce server administrator.</FONT>
             <P>SQL Code = $(SQL_CODE), SQL State = $(SQL_STATE)</BODY></HTML>
        %} : exit
    %}
%}

%HTML_REPORT{
<!-- ensure that user has permission to use this page -->
@AbortIfAdminAccessDeniedForID(SESSION_ID)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<TITLE>Manage Gift Message/Wrapping Feature</TITLE>
</HEAD>
```

```
<BODY BGCOLOR="#FFFFFF">
<!-- fetch informations about the merchant -->
@GetGiftInformations()

<H2>Gift Message/Wrapping</H2>
%IF(ENABLED=="yes")
<FORM ACTION="input">
<INPUT TYPE="hidden" NAME="merfnbr" VALUE="$(merfnbr)">
<b>Status for Gift Message:</b><BR>
<INPUT TYPE="radio" NAME="gmenabled" VALUE="1" $(enabledInStore)>Enabled<BR>
<INPUT TYPE="radio" NAME="gmenabled" VALUE="0" $(disabledInStore)>Disabled<BR>
<b>Current info text</b> (max. 254 chars): <BR>
<TEXTAREA name="gmdesc" COLS=48 ROWS=4 MAXLENGTH="16" WRAP>$(gmdesc)</TEXTAREA><BR>
<INPUT TYPE="submit" VALUE="UPDATE">
</FORM>

<H3>Wrapping preview</H3>
<b>1</b><IMG SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap1.gif" BORDER="0">
 <b>2</b><IMG SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap2.gif" BORDER="0">
 <b>3</b><IMG SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap3.gif" BORDER="0">
 <b>4</b><IMG SRC="/@DTW_rstrip(LongStoreName)/images/giftwrap4.gif" BORDER="0">
<P>To upload new preview images, select the <b>Store Style/Manage Files</b>

%ELSE
<b>The feature is NOT enabled for your store</b>, please contact your ISP.
%ENDIF
</BODY>
</HTML>
%}

%FUNCTION(DTW_ODBC) UpdateGiftMessage() {
    UPDATE
        giftmessage
    SET
        gmenabled = $(gmenabled),
        gmdesc = '$(gmdesc)'
    WHERE
        gmmenbr = $(merfnbr)
%}

%HTML_INPUT{
@UpdateGiftMessage()
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
<TITLE>Manage Gift Message/Wrapping Feature</TITLE>
</HEAD>
<BODY onLoad="this.location.href='report?merfnbr=$(merfnbr)'">
<H2>Updating status for Gift Message</H2>
</BODY>
</HTML>
%}
```

G.5 Makefile for addGiftMsg OF

```
##
#############################################################################################
##
## FILE NAME:    makefile
##
## DESCRIPTION:  makefile to build the ITSO lib for Net.Commerce
```

```
##

GCC              = xlC_r
LINK             = /usr/lpp/xlC/bin/linkxlC_r
LINKSHR          = /usr/ibmcxx/bin/makeC++SharedLib_r
MKLIB            = ar -v -r -u
INCLUDE          = -I/usr/lpp/NetCommerce3/adt/include
CFLAGS           = $(DEBUG) -w -DAIX -qlistopt -qlist -qxref=full -c $(INCLUDE)
GCAPILIBFLAGS    = -+ $(DEBUG) -DAIX -DNLS_ENABLED -DCLNK #-DSYSCTRL_UNIX

LIBS             = -L./ -L/usr/lpp/xlC/lib -lld -L/usr/lpp/NetCommerce3/adt/lib \
       -lC -lnc3_dbc \
       -lserver_objs \
       -lnc3_messages \
       -lnc3_common \
       -lnc3_containers \

LIBPATH          = .

SONAME           = libITSO.a

OBJS      =addGiftMsg.o

all              : $(OBJS) $(SONAME)

addGiftMsg.o  : addGiftMsg.cpp
           $(GCC) $(CFLAGS) addGiftMsg.cpp

$(SONAME): $(OBJS)
       $(LINKSHR) -bloadmap:libITSO.loadmap -p0 \
                        -o $(SONAME)  $(OBJS) $(LIBS)

clean            :
       @echo -
       @echo -
       @echo == Cleaning... ================
       @echo -
       rm *.o *.lst
       rm $(LIBPATH)/$(SONAME)
```

G.6 Source code for addGiftMsg.cpp

```
//
================================================================================================
//
// Licensed Materials - Property of IBM
//
// (c)  Copyright  IBM Corp. 1995, 1997, 1998, 1999, 2000.      All Rights Reserved
//
// US Government Users Restricted Rights - Use, duplication or
// disclosure restricted by GSA ADP Schedule Contract with IBM Corp
//
// addGiftMsg.cpp - Overridable function for task EXT_ORD_PROC
//
// Implicit input parameters:
// ORDER_REF_NUM  : The reference number of the order
//
// This task updates the status of the current order. The fields that is updated are:
// ORFIELD1, ORFIELD3 in table ORDERS, depending on weather the parameters
```

```
// 'giftmsg' & 'wrapping' was given to the OrderProcess command.
//
//
//============================================================================================
#include "nc_core.pch"
#include "objects/objects.pch"

#if defined (WIN32)
#define __DLL_NCS_API__ __declspec(dllexport)
#elif defined(AIX)
#define __DLL_NCS_API__
#endif

//============================================================================================
#ifdef __TRACE_NCAPIS__
typedef TraceYes Trace;
#else
typedef TraceNo  Trace;
#endif
static Trace trace("NC_APIS ("__FILE__")");
//============================================================================================

class __DLL_NCS_API__ addGiftMsg : public NC_OverridableFunction
{
  static const ClassName _STR_ThisClass;

public:
  addGiftMsg() {
    Trace::Tracer T(_STR_CONSTRUCTOR, _STR_ThisClass);
  }

  virtual ~addGiftMsg() {
    Trace::Tracer T(_STR_DESTRUCTOR, _STR_ThisClass);
  }

  void  operator delete( void* p ) { ::delete p; }

public:
   virtual bool Process(const HttpRequest& Req, HttpResponse& Res, NC_Environment& Env)
{
       Trace::Tracer T("ProcessGiftMsg", _STR_ThisClass);

       //
*****************************************************************************
       // the parameter ORDER_REF_NUM is defined by the Net.Commerce API
       //
       const String& OrderRefNum = *(const String *)Env.Seek("ORDER_REF_NUM");
       long ordernumber = 0;
       long merchantnumber = 0;
       OrderRefNum.getVal(ordernumber);
       _MerchantRefNum.getVal(merchantnumber);

       //
*****************************************************************************
       // the parameters giftmsg & wrapping is posted by the form
       // both could come empty...
       //
       const StringWithOwnership NVP_giftmessage("giftmsg");
       const StringWithOwnership NVP_wrapping("wrapping");

       const NameValuePairMap& NVPMap = Req.getNVPs();
       const String& GiftMessage = NVPMap.Get(NVP_giftmessage);
       const String& Wrapping = NVPMap.Get(NVP_wrapping);
```

```
        if(GiftMessage.IsEmpty()) {
            trace << indent << "ProcessGiftMsg: No gift Message for order_rn = " <<
OrderRefNum << endl;
        }

        if(Wrapping.IsEmpty()) {
            trace << indent << "ProcessGiftMsg: No Wrapping for order_rn = " <<
OrderRefNum << endl;
        }

        if(GiftMessage.IsEmpty() && Wrapping.IsEmpty()) {
            trace << indent << "ProcessGiftMsg: Nothing to update..." << endl;
            return true;
        }

        //
*******************************************************************************
        // add the gift message and wrapping to the current order, if they exist
        //
        String stmt;
        DataBase *DbConnection = DataBaseManager::GetCurrentDataBase();

        stmt.Clean();
        stmt << "UPDATE ORDERS SET ";
        if (!GiftMessage.IsEmpty()) {
            stmt << "ORFIELD3 = '" << GiftMessage << "' ";
            if(!Wrapping.IsEmpty())
                stmt << ", ";
        }
        if (!Wrapping.IsEmpty()) {
            stmt << "ORFIELD1 = " << Wrapping << " ";
        }

        stmt << "WHERE ORRFNBR = " << ordernumber;

        SQL SqlUpdateOrders(*DbConnection, stmt);

        if (SqlUpdateOrders.getSQLrc() != ERR_DB_NO_ERROR) {
            SqlUpdateOrders.ReportError();
            trace << indent << "ProcessGiftMsg : Updating orders failed, ORRFNBR = " <<
OrderRefNum << endl;
        } else {
            trace << indent << "ProcessGiftMsg : Updating orders successful, ORRFNBR = "
<< OrderRefNum << endl;
        }

        return true;
    }
};


const ClassName addGiftMsg::_STR_ThisClass("addGiftMsg");
static bool X = NC_OverridableFunctionManager::GetUniqueInstance().RegisterApi
("IBM ITSO", "WCS SPE", "addGiftMsg", 1.0, new addGiftMsg);

static String Copyright = "(c) Copyright IBM Corp. 1995, 1997, 1998, 1999, 2000. All
Rights Reserved";
```

G.7 Modified OrderDetails.tem template

```
Model env, nchs(orderMgmtNLS), merchant, parameters, homeDirectory

Var nameSeparator
Var orderNo = parameters.selectedOrders|toInteger()

Var orderHome = homeDirectory.lookup("OrderHome")
Var order = orderHome.find(orderNo)

Var orderCommentsHome = homeDirectory.lookup("OrderCommentsHome")
Var comments = orderCommentsHome.findByOrder(orderNo)
Var defaultPaymentUrl, customPaymentUrl
Var linkPaymentEnabled = "false"
Var cyberSize = 0
Var cyberCardNumber, cyberState = "", cyberStatus


display()
{
  -- for cybercash
   Query cyber
   cyber = "SELECT CYSTATE  AS STATE,"
           + "CYSTATUS AS STATUS, "
           + "CYCARD_NUMBER AS CARDNUMBER "
           + "FROM CYPAYMTHD "
           + "WHERE CYORNBR=" + parameters.selectedOrders

    cyber|reset()
    cyberSize = cyber.size()
    if ( cyberSize != 0 )
    {
       cyberCardNumber = cyber.cardnumber
       cyberState      = cyber.state
       cyberStatus     = cyber.status
    }

  --
  -- nameSeparator varies depending on locale
  --
  nameSeparator = ", "
  if (env.locale|upperCase() == "JA_JP" || env.locale == "fr_FR")  nameSeparator = "  "

  --
  -- check the payment configuration xml and load the required data
  --
  Model xmlDirectory
  Var paymentConfig = xmlDirectory.lookup("paymentConfig.xml").read(true)
  Var paymentLink = paymentConfig.settings.link
  repeat ( paymentLink )
  {
    if ( paymentLink.name == "paymentProcessing" )
    { -- we found the payment processing tag in payment configuration xml
      defaultPaymentUrl = paymentLink.defaultUrl
      customPaymentUrl  = paymentLink.customUrl
      linkPaymentEnabled= paymentLink.linkEnabled
    }
  }


/*
<HTML>
```

```
<!--============================================================================
Licensed Materials - Property of IBM

5648-B47

(c)  Copyright  IBM Corp.  1998, 1999, 2000.      All Rights Reserved

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
============================================================================-->
<HEAD>
  <LINK REL="stylesheet" HREF="/NCTools/html/common/pagestyle.css">
  <STYLE>
    H2 {font-size:14pt}
    .selectWidth {width: 90px;}
  </STYLE>
</HEAD>

<SCRIPT SRC="/NCTools/javascript/Util.js"></SCRIPT>
<SCRIPT>
  //
  // This function launches the window for adding comments to an order.
  //
  function launchChangeStatus()
  {
    if ( "$order.status$" != "C" && "$order.status$" != "N" &&
         "$order.status$" != "A" && "$order.status$" != "M" &&
         "$order.status$" != "P" && "$order.status$" != "F" ) {
       alert ( "$orderMgmtNLS.orderProcessValidStatesForChangeStatus$" );
       return;
    }

    var now = new Date();
    parent.location.href = 'http://$env.hostname$/servlet/MerchantAdmin?'
                    + 'DISPLAY=CTnchs.order_mgmt.ChangeStatus'
                    + '&selectedOrders=$parameters.selectedOrders$'
                    + '&sortType=$parameters.sortType$'
                    + '&showType=$parameters.showType$'
                    + '&currentOrderState=$parameters.currentOrderState$'
                    + '&CTS=' + now.getTime();
  }

  //
  // This function returns to the order process page
  //
  function returnToOrderProcess()
  {
    var now = new Date();
    parent.location.href = 'https://$env.hostname$/servlet/MerchantAdmin?'
                    + 'DISPLAY=CTnchs.order_mgmt.Main'
                    + '&showType=$parameters.showType$'
                    + '&sortType=$parameters.sortType$'
                    + '&CTS=' + now.getTime();
  }


  //
  // This function removes the selected comments from an order.
  //
  function launchRemoveComments()
  {
    var now = new Date();
    var commentsToRemove = "";
```

```
    var numOfRemovableComments = 0;

    for ( var i = 0; i < $comments.size()$; i++ )
    {
      if(defined(document.orderDetails["comment"+i]))
      {
        numOfRemovableComments++;

        if ( document.orderDetails["comment"+i].checked )
        {
          if ( commentsToRemove != "" )
             commentsToRemove += ";"

          commentsToRemove += document.orderDetails["comment"+i].value
        }
      }
    }

    if(numOfRemovableComments == 0) {
      alert( "$orderMgmtNLS.orderDetailsNoCommentsToRemove$" );
    }else if ( (commentsToRemove == "") && (numOfRemovableComments > 0) ) {
       alert( "$orderMgmtNLS.orderDetailsPleaseSelectComment$" );
    }else {
       if (confirm("$orderMgmtNLS.orderDetailsRemoveCommentConfirm$")) {
          window.location = '/servlet/MerchantAdmin'
                          + '?PROCESS=CTnchs.order_mgmt.OrderDetails'
                          + '&DISPLAY=CTnchs.order_mgmt.OrderDetails'
                          + '&selectedOrders=$parameters.selectedOrders$'
                          + '&currentOrderState=$parameters.currentOrderState$'
                          +
'&showType=$parameters.showType$&sortType=$parameters.sortType$'
                          + '&removeComments=' + escape(commentsToRemove)
                          + '&CTS=' + now.getTime();
       }
    }
  }

  /**
  *  This function launches the window for payment server defined in XML file
  *  only one order will be processed at a time
  *~/
 function launchPaymentServer(paymentServerURL)
  {
    var now = new Date();
    var url = paymentServerURL;

    if ( url.substring(0, 4) == "http" )
    { // this is a directly html launch, not a servlet or CGI command
      var a = window.open( url
                         , "",
"width=750,height=550,screenX=150,screenY=100,resizable=yes,scrollbars=yes");
    }
    else
    { // this is to launch a CGI or servlet command
       var a = window.open( 'http://$env.hostname$'
                  + url
                  + '&order_rn='
                  + '$parameters.selectedOrders$'
                  + '&CTS='
                  + escape(now)
                  , "",
"width=750,height=550,screenX=150,screenY=100,resizable=yes,scrollbars=yes");
    }
```

```
  }

  function launchEditOrderWizard()
  {
   if( ("$parameters.currentOrderState$" != "C") && ("$parameters.currentOrderState$" !=
"N") &&
       ("$parameters.currentOrderState$" != "F") )
    {
      if ( "$cyberState$" != "AUTHFAILED" )
      { // allow to edit the cybercash failed order
        alert("$orderMgmtNLS.orderDetailsCantEditOrderMsg$");
        return;
      }
    }

    var url =
"https://$env.hostname$/servlet/MerchantAdmin?DISPLAY=CTorder_mgmt.order_plcmt.Model" +
              "&XMLFile=order_mgmt.editOrderWizard.xml" +
              "&orderNumber=$order.refno$" +
              "&showType=$parameters.showType$" +
              "&sortType=$parameters.sortType$";

    // if a store is a basic store, we disable the mshipmode page from orderPlacment
wizard
    if ( "$merchant.level|trim()$" == "Basic" )
    {
       url += "&disablePanel=selectShipModeTab";
    }

    var features =
",width=760,height=560,resizable=yes,status=1,scrollbars=yes,menubar=no,copyhistory=no";

    var a = openWindow(url, "EditOrder", null, features);

    parent.location.href = "/NCTools/html/prod_id.html";
    // focus function is not working under IE4 with Service pack 2
    // it will crash, we need to detect the specific IE version 'MSIE 4.01' - d48999
    // we only use focus() function for all browser except IE 4.01
    var navVersion = navigator.appVersion
    if ( navVersion.indexOf("MSIE 4.01") == -1 )
      a.focus();
  }

</SCRIPT>

<body>
<FORM NAME="orderDetails">
  <H2>$orderMgmtNLS.orderDetailsTitle$</H2>*/ endl

  horizontalSeparator(orderMgmtNLS.orderDetailsOrderInformationSeparator)

  --
  -- Display the current order attributes (i.e. OrderNo)
  --
  addOrderInformation()

  --
  -- List the products, tax, shipping, tax on shipping and total charges in
  -- this order
  --
  addProductSummary()

  --
```

```
  -- Add the shipping method
  --
  /* <BR><BR> */
  addShippingMethod()

  --
  -- Add the tax description
  --
  /* <BR><BR> */
  addTaxDescription()

  --
  -- Add shipping & billing addresses
  --
  /* <BR><BR> */
  horizontalSeparator(orderMgmtNLS.orderDetailsAddressInformationSeparator)
  addShippingAddress()

  --
  -- Add the payment information
  --
  horizontalSeparator(orderMgmtNLS.orderDetailsProductPaymentInformationSeparator)
  addPaymentInformation()

  --
  -- Add the order comments
  --
  /* <BR><BR> */
  horizontalSeparator(orderMgmtNLS.orderDetailsCommentSeparator)
  addComments()

  --
  -- Show gift message/wrapping informations
  --
  addGiftMessage()

/*
</FORM></body>
</HTML> */ endl
}

--
-- Display the current order attributes (i.e. OrderNo, Product list)
--
addOrderInformation()
{
  /*<TABLE>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsOrderNumber$</TH>
        <TD>$parameters.selectedOrders$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsLastUpdate$</TH>
        <TD>$order.lastUpdate|format_timestamp()$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsProductOrderState$</TH>
        <TD>$getOrderStatus(parameters.currentOrderState)$</TD>
      </TR>
    </TABLE> */
}

--
```

```
-- Display the items, tax, shipping, tax on shipping and total charges
-- in this order
--
addProductSummary()
{
  Var totalTax         = format_number(order.totalTax)
  Var totalShipping    = format_number(order.totalShipping)
  Var totalShippingTax = format_number(order.totalShippingTax)
  Var totalPrice       = format_number(order.totalPrice)
  Var grandTotal

  grandTotal = totalPrice|add(totalTax)|add(totalShipping)|add(totalShippingTax)

  Query itemList
  --
  -- Query for item list information
  --
  itemList = "SELECT STPRICE  AS PRICE,"
             + "STQUANT  AS QUANTITY,"
             + "PRSDESC  AS DESCRIPTION,"
             + "PRNBR    AS SKU,"
             + "STCMT    AS COMMENTS "
             + "FROM SHIPTO, PRODUCT "
             + "WHERE STORNBR=" + orderNo
             + " AND PRRFNBR=STPRNBR"
  repeat(itemList) {}

  /*
  <CENTER>
  <TABLE CELLPADDING=2 CELLSPACING=0>
    <TR ALIGN=CENTER BGCOLOR="#FBE7A2">
      <TH>$orderMgmtNLS.orderDetailsQuantity$</TH>
      <TH>$orderMgmtNLS.orderDetailsInvoiceProduct$</TH>
      <TH>$orderMgmtNLS.orderDetailsProductDesc$</TH>
      <TH>$orderMgmtNLS.orderDetailsProductComment$</TH>
      <TH>$orderMgmtNLS.orderDetailsProductPrice$ [$merchant.currency|strip()$]</TH>
      <TH>$orderMgmtNLS.orderDetailsProductTotal$ [$merchant.currency|strip()$]</TH>
    </TR> */ endl

      --
      -- List the products contained in this order
      --
      Var itemCost
      Var counter = new Counter(0,1)
      repeat (itemList, counter) {
         if ( counter % 2 == 0 ) { /* <TR BGCOLOR="#EFF3F6"> */ }
         else { /* <TR BGCOLOR="#DDE3F2"> */ }

         itemCost =
itemList.price|multiply(itemList.quantity)|valueSET_HTML(merchant.currency,merchant.seco
ndaryCurrency)

         /* <TD VALIGN=TOP ALIGN=CENTER>$itemList.quantity$</TD>
            <TD VALIGN=TOP>$itemList.description$</TD>
            <TD VALIGN=TOP>$format_string(itemList.sku)$</TD>*/endl

         if(itemList.comments|trim() == "null")
         {
         /* <TD> </TD>*/endl
         }
         else
         {
         /* <TD VALIGN=TOP>$format_string(itemList.comments)$</TD>*/endl
```

```
         }
         /* <TD VALIGN=TOP
ALIGN=RIGHT><NOBR>$itemList.price|valueSET_HTML(merchant.currency,merchant.secondaryCurr
ency)$</NOBR></TD>
            <TD VALIGN=TOP ALIGN=RIGHT><NOBR>$itemCost$</NOBR></TD>
            </TR> */ endl
      }
/*    <TR>
      <TD NOWRAP ALIGN=RIGHT COLSPAN=4>$orderMgmtNLS.orderDetailsProductTax$</TD>
      <TD></TD>
      <TD ALIGN=RIGHT><NOBR>$totalTax|valueSET_HTML(merchant.currency,
merchant.secondaryCurrency)$</NOBR></TD>
    </TR>
    <TR>
      <TD NOWRAP ALIGN=RIGHT COLSPAN=4>$orderMgmtNLS.orderDetailsProductShipping$</TD>
      <TD></TD>
      <TD ALIGN=RIGHT><NOBR>$totalShipping|valueSET_HTML(merchant.currency,
merchant.secondaryCurrency)$</NOBR></TD>
    </TR>
    <TR>
      <TD NOWRAP ALIGN=RIGHT
COLSPAN=4>$orderMgmtNLS.orderDetailsProductTaxOnShipping$</TD>
      <TD></TD>
      <TD ALIGN=RIGHT><NOBR>$totalShippingTax|valueSET_HTML(merchant.currency,
merchant.secondaryCurrency)$</NOBR></TD>
    </TR>
    <TR>
      <TD NOWRAP ALIGN=RIGHT COLSPAN=4><B>$orderMgmtNLS.orderDetailsTotalForOrder$
[$merchant.currency|strip()$]</B></TD>
      <TD></TD>
      <TD ALIGN=RIGHT><NOBR><B>$grandTotal|currencySET_HTML(merchant.currency,
merchant.secondaryCurrency)$</B></NOBR></TD>
    </TR>
  </TABLE>
  </CENTER>*/ endl
}

--
-- Print the shipping method if one exists
--
addShippingMethod()
{
  Var shipToHome = homeDirectory.lookup("ShipToHome")
  Var shipTo = shipToHome.findByOrder(orderNo)
  Var shipMethod

  if (shipTo.mshipModeRefno == "0") {
     shipMethod = orderMgmtNLS.orderDetailsNoShipModeSelected
  } else {
     Var shipModeHome = homeDirectory.lookup("ShipModeHome")
     Var shipMode     = shipModeHome.findByMshipMode(shipTo.mshipModeRefno|toInteger())
     shipMethod       = shipMode.carrier|strip() + ", " + shipMode.mode|strip()
  }

  /* <B>$orderMgmtNLS.orderDetailsShipMode$</B> $shipMethod$ */
}

addTaxDescription()
{
  Var taxJurstHome = homeDirectory.lookup("TaxJurisdictionHome")
  Var taxRuleHome  = homeDirectory.lookup("TaxRuleHome")
  Var shipToHome   = homeDirectory.lookup("ShipToHome")
```

```
  Var shippingAddress = order.shippingAddress
  Var shipTo          = shipToHome.findByOrder(order.refno)

  Var taxRule
  Var taxJurst = taxJurstHome.findByJurst(shippingAddress.country|trim(),
                                          shippingAddress.state|trim(),
                                          merchant.refno)

  Var productHome = homeDirectory.lookup("ProductHome")
  Var products = productHome.findByOrder(order.refno)

  Var taxDescriptions = ""

  repeat (products) {
     if(taxJurst != "") {
        taxRule = taxRuleHome.findTaxRule( shippingAddress.country|trim()
                                         , shippingAddress.state|trim()
                                         , products.taxCode|trim()
                                         , merchant.refno)
        if( taxRule != "" && taxRule.description != "" ) {
           if ( taxRule.description|trim() == "" )
             taxDescriptions += taxRule.code|trim() + ";"
           else
            taxDescriptions += taxRule.code|trim() + " (" + taxRule.description|trim() +
");"
        }
     } else {
        taxRule = taxRuleHome.findTaxRule( shippingAddress.country|trim()
                                         , ""
                                         , products.taxCode|trim()
                                         , merchant.refno)
        if( taxRule != "" && taxRule.description != "" ) {
           if ( taxRule.description|trim() == "" )
             taxDescriptions += taxRule.code|trim() + ";"
           else
            taxDescriptions += taxRule.code|trim() + " (" + taxRule.description|trim() +
");"
        }
     }
  }

  --
  -- Display the tax descriptions, if any.
  --
  Var taxCode

  if( taxDescriptions == "" )
  {
    taxCode = orderMgmtNLS.orderDetailsNoTaxCodeDescription
  }
  else
  {
    taxCode = taxDescriptions
  }

  /* <B>$orderMgmtNLS.orderDetailsTaxCodeDescription$</B> $taxCode$ */
}

--
-- Add the shipping & billing Information for this order
--
addShippingAddress()
{
```

```
  Var shippingAddress = order.shippingAddress
  Var billingAddress = order.billingAddress


  /* <TABLE COLS=3>
      <TR>
       <TD><B>$orderMgmtNLS.orderDetailsShipTo$</B>
        <TABLE>

<TR><TD>$format_string(shippingAddress.title)$$format_string(shippingAddress.lastName)$$
nameSeparator$$format_string(shippingAddress.firstName)$</TD></TR>
          <TR><TD>$format_string(shippingAddress.address1)$
$format_string(shippingAddress.address2)$
$format_string(shippingAddress.address3)$,</TD></TR>
          <TR><TD>$format_string(shippingAddress.city)$,
$format_string(shippingAddress.state)$</TD></TR>
          <TR><TD>$format_string(shippingAddress.country)$,
$format_string(shippingAddress.zipCode)$</TD></TR>
          <TR><TD>$format_string(shippingAddress.daytimePhone)$</TD></TR>
        </TABLE>
       </TD>

       <TD CLASS='selectWidth' width='15'> </TD>

       <TD VALIGN=TOP><B>$orderMgmtNLS.orderDetailsBillTo$</B>
        <TABLE> */ endl

        -- Add billing address
        if ( billingAddress == shippingAddress ) {
          /* <TR VALIGN=TOP><TD>$orderMgmtNLS.orderDetailsSameAsBillToAddress$</TD></TR>
*/
        } else {
           /*  <TR><TD>$format_string(billingAddress.title)$
$billingAddress.lastName$$nameSeparator$$format_string(billingAddress.firstName)$</TD></
TR>
             <TR><TD>$billingAddress.address1$ $format_string(billingAddress.address2)$
$format_string(billingAddress.address3)$,</TD></TR>
              <TR><TD>$billingAddress.city$, $billingAddress.state$</TD></TR>
              <TR><TD>$billingAddress.country$, $billingAddress.zipCode$</TD></TR>
              <TR><TD>$billingAddress.daytimePhone$</TD></TR> */ endl
        }
        /*
        </TABLE>
       </TD>
      </TR>
     </TABLE> */ endl
}

--
-- Retrieve and format the payment information for display.
--
addPaymentInformation()
{
  Var payMethodHome = homeDirectory.lookup("PaymentMethodHome")
  Var payMethod     = payMethodHome.findByOrder(parameters.selectedOrders|toNumber())

  /*
    <TABLE>
    <TR>
      <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsProductPaymentMethod$:</TH>
  */
  if ( payMethod == ""  ) {
    /*
```

```
        <TD>$orderMgmtNLS.orderDetailsNoCreditCardInfo$</TD>
      </TR>
    */
  }
  else if ( payMethod.method == "CYBER" ) {

    if ( cyberSize != 0 )
    {
      /*
        <TD>$orderMgmtNLS.orderDetailsCybercashName$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCreditCardType$:</TH>
        <TD>$payMethod.method$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCreditCardNumber$:</TH>
        <TD>$cyberCardNumber$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCreditCardExpireDate$:</TH>
        <TD>$payMethod.expiryDate|substring(6, 2)$/$payMethod.expiryDate|substring(0,
4)$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCybercashState$:</TH>
        <TD>$cyberState$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCybercashStatus$:</TH>
        <TD>$cyberStatus$</TD>
      </TR>
      */
    }
  }
  else if ( payMethod.method|substring(1,3) != "SET" ) {

    /*
      <TD>$orderMgmtNLS.orderDetailsOfflinePayment$</TD>
    </TR>
    <TR>
      <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCreditCardType$:</TH>
      <TD>$payMethod.method$</TD>
    </TR>
    <TR>
      <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCreditCardNumber$:</TH>


    */

    if ( payMethod.paymentDevice | strip() | isDigit() == "true" ) {
      /*
        <TD>$payMethod.paymentDevice|strip()$</TD>
      </TR>
      <TR>
        <TH ALIGN=LEFT>$orderMgmtNLS.orderDetailsCreditCardExpireDate$:</TH>
        <TD>$payMethod.expiryDate|substring(6, 2)$/$payMethod.expiryDate|substring(0,
4)$</TD>
      </TR>
      */
    }
    else if (  payMethod.paymentDevice|occurencesOf("OFF-LINE")  == "1"  ) {
```

```
      /*
        <TD>$orderMgmtNLS.orderDetailsNoCreditCardInfo$</TD>
      </TR>
      */
    }
  }
  else if ( payMethod.method|substring(1,3) == "SET" ) {

    /*
      <TD>$payMethod.method$</TD>
    </TR>
    */
  }

  /*</TABLE>*/

}

--
-- Add gift message/wrappings for this order
--
addGiftMessage()
{
    Query stmtgift
    Var   giftmsg
    Var   wrapping

    giftmsg = ""
    wrapping = ""

    stmtgift = "SELECT ORFIELD1, ORFIELD3 FROM ORDERS WHERE ORRFNBR=" +
parameters.selectedOrders

    stmtgift|reset()
    wrapping += stmtgift.orfield1
    giftmsg += stmtgift.orfield3

    if ( giftmsg != "null" ) {
        /*
        <H3>Gift Message:</H3>
        $giftmsg$
        */
    }
    if ( wrapping != "0" && wrapping != "null" ) {
        /*
        <H3>Selected Wrapping</H3>
        Wrapping no. $wrapping$
        */
    }
}

--
-- Add the comments for this order to the form
--
addComments()
{
/*  <TABLE>*/ endl
  if ( comments.size() == 0 )
  {
/*    <TR><TD><FONT
COLOR=RED>$orderMgmtNLS.orderDetailsNoCommentsEntered$</FONT></TD></TR>*/
  }
  else
```

```
  {
/*    <TR><TD
colspan=2>$orderMgmtNLS.orderDetailsRemoveCommentsDescMsg$<BR><BR><BR></TD></TR>*/
    Var commentCounter = new Counter(0,1)
    repeat(commentCounter,comments)
    {
/*    <TR>*/ endl

      if(comments.readonly == true)
      {
/*      <TD> </TD>*/ endl
      }
      else
      {
/*      <TD><INPUT TYPE=checkbox NAME="comment$commentCounter$"
VALUE="$comments.lastUpdate$"></TD>*/ endl
      }

/*      <TD><B>[$comments.lastUpdate|format_timestamp()$]</B><BR>$comments.comment$</TD>
    </TR>*/ endl
    }
  }

/*
   </TABLE>

   <INPUT TYPE="button" VALUE="$orderMgmtNLS.orderDetailsRemoveCommentsButton$"
     onClick="launchRemoveComments()">

   <HR HEIGHT=5 NOSHADE><BR>
*/ endl
}


--
-- Method to create a horizontal separator with the given name
--
horizontalSeparator( name )
{
  /*
    <TABLE WIDTH="100%" BORDER=0>
      <TR>
        <TD><HR HEIGHT=5 NOSHADE></TD>
      </TR>
     <TR>
       <TD><B>$name$</B></TD>
     </TR>
      <TR>
        <TD><HR HEIGHT=5 NOSHADE></TD>
      </TR>
    </TABLE>
  */
}

--
-- converts the status to the locale dependant string
--
getOrderStatus( status  )
{
  if ( status == "C" || status == "N" ) orderMgmtNLS.orderProcessNewStatus
  else if ( status == "X" ) orderMgmtNLS.orderProcessCancelledStatus
  else if ( status == "P" ) orderMgmtNLS.orderProcessPendingStatus
  else if ( status == "M" ) orderMgmtNLS.orderProcessReadyStatus
```

```
  else if ( status == "S" ) orderMgmtNLS.orderProcessShippedStatus
  else if ( status == "A" ) orderMgmtNLS.orderProcessAuthorizedStatus
  else if ( status == "F" ) orderMgmtNLS.orderProcessFailedStatus
}

--
-- formats a string for display
--
format_string(str)
{
  if (str == "null") " "
  else str
}

--
-- formats a number for display
--
format_number(number)
{
  if (number == "null") "0"
  else number
}

--
-- Remove the selected comments from this order
--
process()
{
  if ( !(parameters.removeComments startsWith "UNDEFINED") ) {
     Var removeList = parameters.removeComments|tokenize(";")

     repeat ( removeList ) {

        orderCommentsHome.removeByOrder( parameters.selectedOrders|toInteger(),
removeList|toTimestamp() )
     }
  }
}
```

Appendix H. Provisioning sample code

This appendix contains source code listings for programs used in Chapter 10, “Provisioning” on page 225.

H.1 Provisioning HTML example

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<!--==========================================================================
 Licensed Materials - Property of IBM

 5697-D24

 (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.

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

<head>
<title>Example : Launch Create Store</title>
</head>

<body>
<script Language="JavaScript">
   // setting up the window
   var w;
   var launch_str = "Please be patient. The merchant tool will be launched in another
window.";
   var unsupported_browser_text1 = "The merchant tool requires either:";
   var unsupported_browser_item1 = "Netscape Navigator 4.06 or higher";
   var unsupported_browser_item2 = "Internet Explorer 4.01 or higher";
   var unsupported_browser_text2 = "Install the correct browser before creating a
store.";
   var now = new Date();

   // Declaring the basic & advanced URLs as variables
   var basic =
"/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nchs.store_creat
or.basic.xml&level=Basic&catalogXML=nchs.catalog.simpleCatalog.xml&bannerHTML=registerBa
nnerBasic.html&CTS=" + now.getTime();
   var advanced =
"/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nchs.store_creat
or.advanced.xml&level=Advanced&bannerHTML=registerBanner.html&CTS=" + now.getTime();

   function create_basic(){
    create_store(basic)
   }

   function create_advanced(){
    create_store(advanced)
   }

   function create_store(storeurl)
   {
      if ((navigator.appName.indexOf("Netscape")  > -1 &&
parseFloat(navigator.appVersion) >= 4.06) ||
```

```
          (navigator.appName.indexOf("Microsoft") > -1 &&
parseFloat(navigator.appVersion) >= 4.0)) {
         w = window.open(storeurl, "MerchantTool",
"resizable=no,scrollbars=yes,status=yes,width=780,height=550,screenX=0,screenY=0,left=0,
top=0");
      } else {

alert(unsupported_browser_text1+"\n"+unsupported_browser_item1+"\n"+unsupported_browser_
item1+"\n"+unsupported_browser_text2);
      }
   }
</script>

<center>
<font SIZE="4"><BR>
<B>Opening the wizard in current browser window.</B><br>
</font></p>
<a href="javascript:window.location =
'/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nchs.store_creat
or.basic.xml&level=Basic&catalogXML=nchs.catalog.simpleCatalog.xml&bannerHTML=registerBa
nnerBasic.html&CTS=' + Math.ceil(Math.random()*10000);">Click here to create a Basic
Store</a>
<BR><BR>
<a href="javascript:window.location =
'/servlet/MerchantAdmin?GOTO=Banner&body=CTnchs.sc.CreateStore&storeXML=nchs.store_creat
or.advanced.xml&level=Advanced&bannerHTML=registerBanner.html&CTS=' +
Math.ceil(Math.random()*10000);">Click here to create a Advanced Store</a>
<p>================================================<br>
<font SIZE="4"><BR>
<B>Opening the wizard in a new browser window.</B><br>
</font></p>
</p>
<a href="javascript:create_basic()">Create Basic Store</a>
<BR><BR>
<a href="javascript:create_advanced()">Create Advanced Store</a>
<p><br>
</body>
</html>
```

H.2 Registration example HTML

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<!--============================================================================
 Licensed Materials - Property of IBM

 5697-D24

 (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.

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

<head>
<title>Example ISP Register Page</title>
<script>

var now = new Date();
```

```
function homepage_services(){
// go to home page services page
}

function convert_params(form){
form.firstname.value = escape(form.firstname.value);
form.lastname.value = escape(form.lastname.value);
form.add1.value = escape(form.add1.value);
form.add2.value = escape(form.add2.value);
form.city.value = escape(form.city.value);
form.state.value = escape(form.state.value);
form.country.value = escape(form.country.value);
form.zip.value = escape(form.zip.value);
form.phone.value = escape(form.phone.value);
form.fax.value = escape(form.fax.value);
form.email.value = escape(form.email.value);
form.password.value = escape(form.password.value);
}

function basic_e_store_services(form){

convert_params(form);

form.storeXML.value="nchs.store_creator.basic.xml";
form.level.value="Basic";
form.catalogXML.value="nchs.catalog.simpleCatalog.xml";
form.bannerHTML.value="registerBannerBasic.html";
form.CTS.value= now.getTime();
form.submit();
}

function advanced_e_store_services(form){

convert_params(form);

form.storeXML.value="nchs.store_creator.advanced.xml";
form.level.value="Advanced";
form.bannerHTML.value="registerBanner.html";
form.CTS.value= now.getTime();
form.submit();
}



</script>
</head>

<body>
  <TABLE background="/NCTools/images/catalog/topbanner.gif" BORDER=0 CELLSPACING=0
CELLPADDING=0 width=100%>
<TR>
<td><img src="/EXT_pages/images/earth-topo.gif" width=200 height=100></TD>
    <TD VALIGN=CENTER ALIGN="left"><FONT FACE="Arial,Helvetica,sans-serif" SIZE="5"
COLOR=white><center><b><u>No.1 ISP</u></B><br><br>Registration Form</center></FONT></TD>
<td align=right ><img src="/EXT_pages/images/isplogo.gif" width=200 height=100></TD>
   </TR>
  </TABLE>

<p>
<center>
<p><B>
Please fill in your details and then choose which service you wish to register for.
<form method=post action='/servlet/MerchantAdmin'>
```

```
<input type=hidden name=GOTO value="Banner" >
<input type=hidden name=body value="CTnchs.sc.CreateStore" >
<input type=hidden name=storeXML value="" >
<input type=hidden name=level value="" >
<input type=hidden name=catalogXML value="" >
<input type=hidden name=bannerHTML value="" >
<input type=hidden name=CTS value="" >

<table>
<tr><td>Firstname:</td>
<td><input type=text name="firstname" value="Joe" size=25></td></tr>
<tr><td>Lastname:</td>
<td><input type=text name="lastname" value="Bloggs" size=25></td></tr>
<tr><td>Address:</td>
<td><input type=text name="add1" value="4 Main Street" size=25></td></tr>
<tr><td>Address line 2:</td>
<td><input type=text name="add2" value="Newtown" size=25></td></tr>
<tr><td>City:</td>
<td><input type=text name="city" value="Newcity" size=25></td></tr>
<tr><td>State:</td>
<td><input type=text name="state" value="Florida" size=25></td></tr>
<tr><td>Country:</td>
<td><input type=text name="country" value="United States" size=25></td></tr>
<tr><td>Zip:</td>
<td><input type=text name="zip" value="123 56" size=25></td></tr>
<tr><td>Telephone:</td>
<td><input type=text name="phone" value="1 8 102345" size=25></td></tr>
<tr><td>Fax:</td>
<td><input type=text name="fax" value="1 8 654832" size=25></td></tr>
<tr><td>Email:</td>
<td><input type=text name="email" value="joe@joe.com" size=25></td></tr>
<tr><td>Password:</td>
<td><input type=password name="password" value="joe" size=25></td></tr>
</table>
<table>
<tr><td><p></td></tr>
<tr><td><input type=button value="Home Page Services" onclick="homepage_services()"></td>
<td><input type=button value="E-Store Basic Services"
onclick="basic_e_store_services(this.form)"></td>
<td><input type=button value="E-Store Advanced Services"
onclick="advanced_e_store_services(this.form)"></td>
</tr>
</table>
</B>
</form>
</center>
</font></p>
</body>
</html>
```

Appendix I. Multi-language samples

This appendix contains source code listings for programs used in Chapter 11, “Multi-language support” on page 237.The code is sending an extra XML parameter to the final creation process in the store wizard or store creator panel.

I.1 displaylangs.html

```
<html><head>
<title>DisplayPages</title>
<script>

var enXML = "<?xml version=1.0 encoding=iso-8859-1
?><navigationBar><item><name>Home</name><link><type>home</type><description>homeDesc</de
scription><url>http://ausres20.itsc.austin.ibm.com/$merchant.storeName|strip()$/homepage
.html</url><urlModifiable>true</urlModifiable><urlSwitch>on</urlSwitch></link></item><it
em><name>Catalog</name><link><type>catalog</type><description>catalogDesc</description><
url>http://$env.hostname$/servlet/Catalog?merchant.refno=$merchant.refno$</url><urlModif
iable>true</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><name>Search</nam
e><link><type>search</type><description>searchDesc</description><url>http://$env.hostnam
e$/servlet/StoreSearch?merchant.refno=$merchant.refno$</url><urlModifiable>false</urlMod
ifiable><urlSwitch>on</urlSwitch></link></item><item><name>Shopping
Cart</name><link><type>shoppingCart</type><description>shoppingCartDesc</description><ur
l>http://$env.hostname$/servlet/ViewShoppingCart?merchant.refno=$merchant.refno$</url><u
rlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><name>Cus
tomer
service</name><link><type>customerService</type><description>customerServiceDesc</descri
ption><url>http://$env.hostname$/servlet/CustomerService?merchant.refno=$merchant.refno$
</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><
name>Registration</name><link><type>registration</type><description>registrationDesc</de
scription><url>http://$env.hostname$/servlet/Register?merchant.refno=$merchant.refno$</u
rl><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><nam
e>Log
on</name><link><type>logon</type><description>logonDesc</description><url>http://$env.ho
stname$/servlet/Logon?merchant.refno=$merchant.refno$&amp;merchant.homeURL=$merchant.hom
eURL$</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><i
tem><name>Log
off</name><link><type>logoff</type><description>logoffDesc</description><url>http://$env
.hostname$/servlet/Logoff?merchant.refno=$merchant.refno$&amp;merchant.homeURL=$merchant
.homeURL$</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></ite
m></navigationBar>";

var svXML = "<?xml version=1.0 encoding=iso-8859-1
?><navigationBar><item><name>Hem</name><link><type>home</type><description>homeDesc</des
cription><url>http://ausres20.itsc.austin.ibm.com/$merchant.storeName|strip()$/homepage.
html</url><urlModifiable>true</urlModifiable><urlSwitch>on</urlSwitch></link></item><ite
m><name>Katalog</name><link><type>catalog</type><description>catalogDesc</description><u
rl>http://$env.hostname$/servlet/Catalog?merchant.refno=$merchant.refno$</url><urlModifi
able>true</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><name>Sök</name><l
ink><type>search</type><description>searchDesc</description><url>http://$env.hostname$/s
ervlet/StoreSearch?merchant.refno=$merchant.refno$</url><urlModifiable>false</urlModifia
ble><urlSwitch>on</urlSwitch></link></item><item><name>Kundvagn</name><link><type>shoppi
ngCart</type><description>shoppingCartDesc</description><url>http://$env.hostname$/servl
et/ViewShoppingCart?merchant.refno=$merchant.refno$</url><urlModifiable>false</urlModifi
able><urlSwitch>on</urlSwitch></link></item><item><name>Kund
service</name><link><type>customerService</type><description>customerServiceDesc</descri
ption><url>http://$env.hostname$/servlet/CustomerService?merchant.refno=$merchant.refno$
```

```
</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><
name>Registrera</name><link><type>registration</type><description>registrationDesc</desc
ription><url>http://$env.hostname$/servlet/Register?merchant.refno=$merchant.refno$</url
><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><name>
Logga
på</name><link><type>logon</type><description>logonDesc</description><url>http://$env.ho
stname$/servlet/Logon?merchant.refno=$merchant.refno$&amp;merchant.homeURL=$merchant.hom
eURL$</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><i
tem><name>Logga
av</name><link><type>logoff</type><description>logoffDesc</description><url>http://$env.
hostname$/servlet/Logoff?merchant.refno=$merchant.refno$&amp;merchant.homeURL=$merchant.
homeURL$</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item
></navigationBar>";

var deXML = "<?xml version=1.0 encoding=iso-8859-1
?><navigationBar><item><name>Home</name><link><type>home</type><description>homeDesc</de
scription><url>http://ausres20.itsc.austin.ibm.com/$merchant.storeName|strip()$/homepage
.html</url><urlModifiable>true</urlModifiable><urlSwitch>on</urlSwitch></link></item><it
em><name>Catalog</name><link><type>catalog</type><description>catalogDesc</description><
url>http://$env.hostname$/servlet/Catalog?merchant.refno=$merchant.refno$</url><urlModif
iable>true</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><name>Search</nam
e><link><type>search</type><description>searchDesc</description><url>http://$env.hostnam
e$/servlet/StoreSearch?merchant.refno=$merchant.refno$</url><urlModifiable>false</urlMod
ifiable><urlSwitch>on</urlSwitch></link></item><item><name>Shopping
Cart</name><link><type>shoppingCart</type><description>shoppingCartDesc</description><ur
l>http://$env.hostname$/servlet/ViewShoppingCart?merchant.refno=$merchant.refno$</url><u
rlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><name>Cus
tomer
service</name><link><type>customerService</type><description>customerServiceDesc</descri
ption><url>http://$env.hostname$/servlet/CustomerService?merchant.refno=$merchant.refno$
</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><
name>Registration</name><link><type>registration</type><description>registrationDesc</de
scription><url>http://$env.hostname$/servlet/Register?merchant.refno=$merchant.refno$</u
rl><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><item><nam
e>Log
on</name><link><type>logon</type><description>logonDesc</description><url>http://$env.ho
stname$/servlet/Logon?merchant.refno=$merchant.refno$&amp;merchant.homeURL=$merchant.hom
eURL$</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></item><i
tem><name>Log
off</name><link><type>logoff</type><description>logoffDesc</description><url>http://$env
.hostname$/servlet/Logoff?merchant.refno=$merchant.refno$&amp;merchant.homeURL=$merchant
.homeURL$</url><urlModifiable>false</urlModifiable><urlSwitch>on</urlSwitch></link></ite
m></navigationBar>";


function Initialize(){
   top.pageAlreadyLoaded = true;
   setuppage();
   setselected(document.forms[0]);
   document.forms.f1.merchant_rn.value=top.merchantref;
}

//theOptions variable holds the different lang choices for the user
var theOptions = new Array();

function CreateOptions(lang,XML ) {
this.lang=lang;
this.XML=XML;
return this;
}

//This function prints out each choice on the panel and sets it's values in variable
theOptions.
```

```
function setuppage()
{
theOptions[0]=new CreateOptions('en_US',enXML);
theOptions[1]=new CreateOptions('sv_SE',svXML);
theOptions[2]=new CreateOptions('de_DK',deXML);
}

//When a merchant chooses a page the setXMLChanges fn is called
//to create or update that choice in the newElements array
function storesettings(index){
document.forms.f1.lang.value=theOptions[index].lang;
top.setLang(theOptions[index].lang, theOptions[index].XML);
}

// These functions display the previously selected option if the merchant is returning
// to this page or sets the default selected one.

function setselected(form)
{
var chosen=0;
for (var i in form.choice){
    if (was_selected(theOptions[i].lang)){
        chosen=i;
        break;}
}
form.choice[chosen].checked = true
}

function was_selected(svalue){
var b = false;
var index=0;
while( top.langElements[index] && !b){
if (top.langElements[index].lang == svalue)
    {b=true;break;}
index++;
}
return b;
}


</script>
</head>
<body onLoad="Initialize()">
<p><B><font size=5>
Please choose a language from below:</font></B><p>

<form name=f1 action="/cgi-bin/ncommerce3/langAdd">
<input type=hidden name="merchant_rn" value==""">
<input type=hidden name="lang" value==""">
<input type=hidden name="url"
value=="/cgi-bin/ncommerce3/ExecMacro/displaylangs.d2w/report">
<table border="0" CELLPADDING=2 CELLSPACING=2><tr>

<td><p></td><td><input type=radio name=choice value="0"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2>
<tr><td><font face=Arial, Helvetica, sans-serif>Default - US
English</font></td></tr></table></td>

<td><p></td><td><input type=radio name=choice value="1"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2>
<tr><td><font face=Arial, Helvetica, sans-serif>Swedish</font></td></tr></table></td>
```

```
<td><p></td><td><input type=radio name=choice value="2"
onclick=storesettings(this.value)></td>
<td><table Border=0 CELLPADDING=2 CELLSPACING=2>
<tr><td><font face=Arial, Helvetica, sans-serif>Danish</font></td></tr></table></td>

<td><font face="Arial, Helvetica, sans-serif"><input type="submit" name="assign"
value="Assign language"></font></td></tr>
</table>
</form>
</body>
</html>
```

I.2 langs.js

```
/**************************************************************************
** Licensed Materials - Property of IBM                                  **
**                                                                       **
** 5697-D24                                                              **
**                                                                       **
** (C) Copyright IBM Corp. 1995, 1999.    All Rights Reserved.           **
**                                                                       **
** US Government Users Restricted Rights - Use, duplication or disclosure   **
** restricted by GSA ADP Schedule Contract with IBM Corp.                **
**************************************************************************/

// arrays for changed Elements
var langElements = {lang:'',XML:''};

function setLang( lang, XML ) {
langElements.lang=lang;
langElements.XML=XML;
return this;
}
```

I.3 Changes to Model.tem

```
<SCRIPT SRC="/NCTools/javascript/Vector.js"></SCRIPT>
<SCRIPT SRC="/EXT_NCTools/javascript/lang.js"></SCRIPT>

<SCRIPT>
var merchantref = "$env.merchant_id$";
```

I.4 Changes to advanced.xml

```
<panel name="storeCreatorPanelLanguage"
        url="/EXT_pages/html/displaylangs.html?dummy=test" />
```

I.5 langAdd.cpp

```
/*
############################################################################

langAdd Command for redbook
```

```
Purpose : Wrapper for RegisterNew cmd that relates new shopper to merchant

Usage: /cgi-bin/ncommerce3/langAdd?merchant_rn=<>&lang=<>&url=<>

History
=======
  Version  Date      Who  Comment
  -------  --------  ---  -------


########################################################################
*/

#include "objects/objects.pch"

//=========================================

#ifdef __TRACE_COMMANDS__
  typedef TraceYes Trace;
#else
  typedef TraceNo  Trace;
#endif
static Trace trace("MY_COMMANDS ("__FILE__")");
//=========================================


#if defined(WIN32)
 #define __EXPORT_MODE__ __declspec(dllexport)
#elif defined (AIX)
 #define __EXPORT_MODE__
#endif


//============================================================================
   // variables used by the Process function
   String merchantRefNum;
   String _VAL_url;
   String _VAL_lang;

   //NVP_names declarations
   static const StringWithOwnership NVP_merchant_rn("merchant_rn");
   static const StringWithOwnership NVP_url("url");
   static const StringWithOwnership NVP_lang("lang");


////////////////////////////////////////////////////////////////////////////////
//////////
//
// langAdd
//
////////////////////////////////////////////////////////////////////////////////
//////////
class __EXPORT_MODE__ langAdd : public NC_Command
{
   static const ClassName _STR_ThisClass;


public:
   langAdd(void)
   {
      // Trace the function call
      Trace::Tracer T(_STR_CONSTRUCTOR, _STR_ThisClass);
```

```
   }


   virtual bool Initialize(void)
   {
      // Trace the function call
      Trace::Tracer T(_STR_Initialize, _STR_ThisClass);

      return true;
   }


   virtual ~langAdd(void)
   {
      // Trace the function call
      Trace::Tracer T(_STR_DESTRUCTOR, _STR_ThisClass);

   }

   void  operator delete( void* p ) { ::delete p; }
   virtual NC_Command* Clone(void) { return new langAdd; }

public:
   virtual void FailedRegistration(NC_RegistrationID& RegID, const ErrorMsg_Reg* Err)
   {
      // Trace the function call
      Trace::Tracer T(_STR_FailedRegistration, _STR_ThisClass);

      if (Err == &_ERR_REG_UNEXPECTED_OBJ)
      {
         error << indent << "ERROR: langAdd not accepted by the manager" << endl;
      }
      else if (Err == &_ERR_REG_DUPE_ID)
      {
         error << indent << "ERROR: langAdd: another command with the same name has
already been registered" << endl;
      }
      else if (Err == &_ERR_REG_OBJ_NOT_FROM_DLL)
      {
         error << indent << "ERROR: langAdd is packaged in a different DLL then the one
registered in the DB" << endl;
      }
      else
      {
         error << indent << "ERROR: langAdd unknown registration error" << endl;
      }

      error << indent << "ERROR: langAdd's registration failed" << endl;
   }


   virtual bool CheckParameters(const HttpRequest& Req, HttpResponse& Res,
                              NC_Environment& Env, NC_Environment& Resources)
   {
      // Trace the function call
      Trace::Tracer T(_STR_CheckParameters, _STR_ThisClass);

      if (Misc::CheckFieldExistence(Req, NVP_merchant_rn, merchantRefNum, false)==false)
      return false;

      debug << indent << "AH - CheckParams merchantRefNum" << merchantRefNum << endl;
```

```
      if (Misc::CheckFieldExistence(Req, NVP_lang, _VAL_lang, false)==false)
      return false;

      debug << indent << "AH - CheckParams lang" << _VAL_lang << endl;

      if (Misc::CheckFieldExistence(Req, NVP_url, _VAL_url, false) == false)
      return false;

      debug << indent << "AH - CheckParams url" << _VAL_url << endl;

      // return success if everything went OK
      return true;
   }



   // *********************************8
   // Process section
   virtual bool Process (const HttpRequest& Req, HttpResponse& Res, NC_Environment& Env)
   {
      // Trace the function call
      Trace::Tracer T(_STR_Process, _STR_ThisClass);

      // dump the nvp
      debug << indent <<
"AH====================================================================" << endl;
      debug << indent << "AH start process" << endl;

      debug << indent << " --- DumpInputStart ---" << endl;
      NameValuePairMap::Iterator I(&Req.getNVPs());
      for (I.Begin(); *I != NULL; ++I)
      {
        debug << indent << (*I)->getName().c_str() << " = ";
        debug << (*I)->getValue().c_str() << endl;
      }
      debug << indent << " --- DumpInputEnd   ---" << endl;


      //====================================================================
//MERCHANT

      DataBase& DB = *(DataBase*)Misc::CheckEnvironmentVariable(Env,
NC_Environment::_VAR_MainDatabase);


      MerchantHome merchant_home( DB );

      Merchant* merchant;

      merchant = merchant_home.Lookup(merchantRefNum);

      merchant->setField1(_VAL_lang);

      debug << indent << "Insert row into merchant" << endl;

      // write to table
      if (merchant->Write() != ERR_DB_NO_ERROR)
      return false;

  //=============================================================================
      // redirect to url
      Res.setLocation(_VAL_url,false);
```

```
      debug << indent << "AH - Process end " << endl;
      debug << indent <<
"AH============================================================================" << endl;

      // destroy objects
      merchant_home.Destroy(merchant);

      return true;
   }

};



// Tracing variable to identify calls to functions for your class.

const ClassName langAdd::_STR_ThisClass("langAdd");


// Automatically instanciate your command and registers it to the name you give it.
// "Vendor" is the name of the company you write this command for.
// "Product" is the name of the product/package this command will be part of
// "CommandName" is the actual name of your command
// 1.0 is the version of this command
static bool X1 = NC_CommandManager::Register("IBM"     ,
                                             "NC"    ,
                                             "langAdd",
                                             1.0          ,
                                             new langAdd
                                            );
//=============================================================================================
```

Appendix J. Using the additional material

This redbook also contains additional material available on the Internet. See the section below for instructions on using or downloading each type of material.

J.1 Locating the additional material on the Internet

The source code in the appendices in this redbook is also available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG245958

Alternatively, you can go to the IBM Redbooks Web site at:

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

Select the **Additional materials** and open the directory that corresponds with the redbook form number.
There are two files in the folder SG245958.ZIP (1.19 MB) and SG245958ekstra.ZIP (26.4 MB) the difference between the two is that SG245958ekstra.ZIP contains the javafulfillment.exe application mentioned in Section 7.1, “Integrating external business system with Net.Commerce” on page 127. Unless you need this application, we recommend that you pick the smaller file, SG245958.ZIP

J.2 Using the Web material

The additional Web material that accompanies this redbook includes the following:

File name	*Description*
SG245958.zip	Appendix B to I code examples in Zipped format

J.2.1 System requirements for downloading the Web material

The following system configuration is recommended for downloading the additional Web material.

Hard disk space:	2 MB minimum
Operating System:	Any
Processor:	Any
Memory:	Any

J.2.2 How to use the Web material

Create a subdirectory (folder) on your workstation and copy the contents of the Web material into this folder and unzip the files. Then install the files according to the description in the chapters of the book. Details may differ depending on your local environment.

Appendix K. Special notices

This publication is intended to help ASPs, ISPs, IBM business partners, and IBM technical employees customize Websphere Commerce Suite Service Provider Edition in an efficient and reliable way to tailor the product for specific needs. The information in this publication is not intended as the specification of any programming interfaces that are provided by Websphere Commerce Suite Service Provider Edition Version 3.2. See the PUBLICATIONS section of the IBM Programming Announcement for Websphere Commerce Suite Service Provider Edition.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee

that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

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

IBM
RS/6000
SP
ThinkPad
Wizard
RISC System/6000
SAA
System/390
WebSphere

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere.,The Power To Manage., Anything. Anywhere.,TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

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

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

Lotus Notes is a registered trademark of Lotus Development Corporation.

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

Appendix L. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

L.1 IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 399.

- *Building e-commerce Solutions with Net.Commerce: A Project Guidebook*, SG24-5417
- *The XML Files: Using XML and XSL with IBM WebSphere V3.0*, SG24-5479
- *WebSphere Application Servers: Standard and Advanced Editions*, SG24-5460

L.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates, and formats.

CD-ROM Title	Collection Kit Number
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

L.3 Other resources

These publications are also useful as further information sources:

- *AIX Version 3.2 Commands Reference, Volume 3*, GC23-2367

- *e-business with Net.Commerce* by Samantha Shurety, Prentice Hall. ISBN 0-1308-3808-x
- *Installing and Getting Started Guide V3.2*, GC09-2808

The following two publications can be found at:
`http://www-4.ibm.com/software/webservers/commerce/servers/lit-tech-general.html`

- *Commands, Tasks, Overridable Functions, and Database Tables*
- *Commands, Tasks, Overridable Functions, and the e-commerce Data Objects*

The following publications are product documentation and must be purchased with the software product:

- *Customization Guide, Version 3.2*
- *Java Fulfillment: Integrating Java With IBM Net.Commerce*

L.4 Referenced Web sites

The following Web sites are also relevant as further information sources:

- `http://www-4.ibm.com/software/webservers/commerce/community/process/refapps.html`
- `http://www-4.ibm.com/software/webservers/commerce/servers/lit-tech-general.html`
- `http://www.software.ibm.com/commerce/net.commerce`
- `http://www.software.ibm.com/webservers/appserv/`
- `http://www.software.ibm.com/data/net.data/`
- `http://www.software.ibm.com/data/db2/`
- `http://www.ibm.com/developer/xml/`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** ibm.com/redbooks

  Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

  Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

  Send orders by e-mail including information from the IBM Redbooks fax order form to:

	e-mail address
In United States or Canada	pubscan@us.ibm.com
Outside North America	Contact information is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: http://www.elink.ibmlink.ibm.com/pbl/pbl

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at http://w3.itso.ibm.com/ and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at http://w3.ibm.com/ for redbook, residency, and workshop announcements.

IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

ACL. Access Control List.

Acquiring institution. The financial institution that receives payment from the customer's financial institution and forwards it to the merchant.

ANSI. American National Standards Institute.

API. Application Programming Interface.

ARP. Address Resolution Protocol.

Application Services Provider. (ASP) An Internet sevice provider that offers application services on the Internat rather than just connection service.

ASP. Application Service Provider. An Internet service provider that offers application services. See ISP.

ATM. Asynchronous Transfer Mode.

BIND. Berkeley Internet Name Daemon.

BOS Base Operating System.

Cache. A special-purpose buffer storage that is used to hold a copy of data that may be frequently accessed. Use of a cache reduces access time but may increase memory requirements.

Caching utility. A utility provided by Commerce Service Provider that enables the user to cache certain frequently-used pages.

Catalog. In Commerce Service Provider, a collection of Web pages that contain information about goods or services offered for sale by a single store.

CDE. Common Desktop Environment.

CDLI. Common Data Link Interface.

CD-R. CD Recordable.

CE. Customer Engineer.

CEC. Central Electronics Complex.

Certificate. In secure communications, a digital document that binds an encryption key to the identity of the certificate owner so that the certificate owner can be authenticated. A certificate is issued by a certifying authority

Certifying Authority. (CA) In secure communications, a trusted third party, such as VeriSign, Inc., or a designated internal authority that issues certificates. See also Certificate.

CGE. Common Graphics Environment.

CGI. Common Gateway Interface.

CHRP. Common Hardware Reference Platform.

CLVM. Concurrent LVM.

COFF. Common Object File Format.

Commerce Service Provider Administrator. A series of interfaces that enables merchants to create and manage malls and stores.

CORBA. Common Object Request Broker.

DAD. Duplicate Address Detection.

DASD. Direct Access Storage Device.

DB2. An IBM relational database management system.

DB2 Extenders. An optional product that enhances the search capabilities of DB2 by providing extensions to SQL. These extensions work with a search engine to allow shoppers to retrieve information from the DB2 database.

DBE. Double Buffer Extension.

DBCS. Double Byte Character Set.

DCE. Distributed Computing Environment.

DES. Data Encryption Standard.

DHCP. Dynamic Host Configuration Protocol.

Distinguished name. In secure communications, the name and address of the person and organization to whom a certificate has been issued. See also Certificate.

DIT. Directory Information Tree.

DMA. Direct Memory Access.

DN. Distinguished Name.

DNS. Domain Naming Server.

Domino Go Webserver. The Web server provided with Commerce Service Provider.

DS. Differentiated Service.

DSE. Diagnostic System Exerciser.

DSMIT. Distributed SMIT.

DTE. Data Terminating Equipment.

EA. Effective Address.

ECC. Error Checking and Correcting.

EIA. Electronic Industries Association.

EMU. European Monetary Union.

EOF. End of File.

ESP. Encapsulating Security Payload.

FCAL. Fibre Channel Arbitrated Loop.

FCC. Federal Communication Commission.

FDDI. Fiber Distributed Data Interface.

FDPR. Feedback Directed Program Restructuring.

FIFO. First In/First Out.

FLASH EPROM. Flash Erasable Programmable Read-Only Memory.

FLIH. First Level Interrupt Handler.

FRCA. Fast Response Cache Architecture.

GAI. Graphic Adapter Interface.

GPR. General Purpose Register.

GUI. Graphical User Interface.

HACMP. High Availability Cluster Multi-Processing.

HCON. IBM AIX Host Connection Program/6000.

HFT. High Function Terminal.

HTML. Hypertext Markup Language. A notation for identifying the components of a document to format it for display on the Web. HTML formats the text, determines input areas on forms, and creates navigational links.

HTTP. HyperText Transfer Protocol. A TCP/IP protocol used to transfer HTTP commands between the server and the Web client.

IAR. instruction Address Register.

IBM. International Business Machines.

ICCCM. Inter-Client Communications Conventions Manual.

ICE. Inter-Client Exchange.

ICElib. Inter-Client Exchange library.

ICMP. Internet Control Message Protocol.

ICSS. Internet Connection Secure Server. The Web server that was provided with Net.Commerce Version 1 and Version 2.

IETF. Internet Engineering Task Force.

IHV. Independent Hardware Vendor.

IIOP. Internet Inter-ORB Protocol.

IJG. Independent JPEG Group.

IKE. Internet Key Exchange.

ILS. International Language Support.

IM. Input Method.

IP address. The unique 32-bit address that specifies the location of each device or workstation in the Internet. For example, 9.41.41.103 is an IP address.

IPL. Initial Program Load.

IPSec. IP Security.

IS. Integrated Service.

ISA. Industry Standard Architecture.

ISAKMP/Oakley. Internet Security Association Management Protocol.

ISNO. Interface Specific Network Options.

ISO. International Organization for Standardization.

ISP. Internet Service Provider. An Internet service provider that offers services on the Internet, such as connection services. See ASP.

ISV. Independent Software Vendor.

ITSO. International Technical Support Organization. The IBM department that writes IBM Redbooks.

I/O. Input/Output.

JDBC. Java Database Connectivity.

JFC. Java Foundation Classes.

JFS. Journaled File System.

Key. In secure communications, an algorithmic pattern that is used by a sender to encrypt messages and by a recipient to decrypt messages.

Key pair. In secure communications, a key pair consists of a public key and a private key. The sender uses the public key to encrypt the message, and the recipient uses the private key to decrypt the message. See also private key and public key.

Key ring. (key ring certificate) In secure communications, a file that contains public keys, private keys, trusted roots, and certificates. See also certificate, public key, private key.

Mall. In Commerce Service Provider, a collection of electronic stores that are jointly managed by the same administrator.

Mass Import utility. A utility that lets a user quickly import product information from a text file into the Commerce Service Provider database rather than completing forms in Commerce Service Provider Administrator one-by-one.

LAN. Local Area Network.

LDAP. Lightweight Directory Access Protocol.

LDIF. LDAP Directory Interchange Format.

LFT. Low Function Terminal.

LID. Load ID.

LP. Logical Partition.

LPP. Licensed Program Products.

LTG. Logical Track Group.

LV. Logical Volume.

LVCB. Logical Volume Control Block.

LVD. Low Voltage Differential.

LVM. Logical Volume Manager.

MBCS. MultiByte Character Support.

MCA. Micro Channel Architecture. The system bus that was used in IBM RS/6000 and IBM PS/2.

MDI. Media Dependent Interface.

Merchant. In Commerce Service Provider, an individual or company that uses Commerce Service Provider to sell goods or services over the Web.

Merchant key. A 16-digit hexadecimal number that the Configuration Manager uses to encrypt passwords.

MII. Media Independent Interface.

MODS. Memory Overlay Detection Subsystem.

MP. Multiple Processor.

MPOA. Multiprotocol Over ATM.

MST. Machine State.

NBC. Network Buffer Cache.

ND. Neighbor Discovery.

NDP. Neighbor Discovery Protocol.

Net.Data. The language used in Commerce Service Provider macros.

Netscape Enterprise Server. A Web server that can be used with Commerce Service Provider instead of Domino Go Webserver.

NFS. Network File System.

NHRP. Next Hop Resolution Protocol.

NIM. Network Installation Management.

NIS. Network Information System.

NL. National Language.

NLS. National Language Support.

NTF. No Trouble Found.

NVRAM. Non-Volatile Random Access Memory.

OACK. Option Acknowledgment.

ODBC. Open DataBase Connectivity.

ODM. Object Data Manager.

OEM. Original Equipment Manufacturer.

OLTP. Online Transaction Processing.

ONC. Open Network Computing.

OOUI. Object-Oriented User Interface.

OSF. Open Software Foundation, Inc.

Overridable function. Program code that implements a task.

PCI. Peripheral Component Interconnect.

PEX. PHIGS Extension to X.

PFS. Perfect Forward Security.

PHB. Processor Host Bridges.

PHY. Physical Layer Device.

PID. Process ID.

PII. Program Integrated Information.

PMTU. Path MTU.

PPC. PowerPC.

Private key. In secure communications, an algorithmic pattern used to decrypt messages that were encrypted by the corresponding public key. You keep your private key on your own system in a key ring, protected by a password.

Product Advisor. A component of the Commerce Service Provider used to create intelligent catalogs.

Production server. The server on which you make your Commerce Service Provider site available to shoppers. See also staging server.

Public key. In secure communications, an algorithmic pattern used to encrypt messages that the corresponding private key can decrypt. You make your public key available to everyone who will need it. See also key ring and private key.

Proxy server. The method for implementing Internet security with a Firewall, Application level proxy or circuit level proxy.

PSE. Portable Streams Environment.

PTF. Program Temporary Fix.

PV. Physical Volume.

QoS. Quality of Service.

RAID. Redundant Array of Independent Disks.

RAN. Remote Asynchronous Node.

RAS. Reliability Availability Serviceability.

RDB. Relational DataBase.

RDISC. ICMP Router Discovery.

RDN. Relative Distinguished Name.

RDP. Router Discovery Protocol.

RFC. Request for Comments.

RIO. Remote I/O.

RIP. Routing Information Protocol.

RPA. RS/6000 Platform Architecture.

RPC. Remote Procedure Call.

RPL. Remote Program Loader.

RSVP. Resource Reservation Protocol.

SA. Secure Association.

SACK. Selective Acknowledgments.

SBCS. Single-Byte Character Support.

SCB. Segment Control Block.

SCSI. Small Computer System Interface.

SCSI-SE. SCSI-Single Ended connection.

SDRAM. Synchronous DRAM.

SE. Single Ended, type of SCSI connection.

Secure server. A server that uses secure protocols to protect the confidentiality of information that is transmitted and received over the Web. Domino Go Webserver is an example of such a server.

Server. A computer that provides shared services to other computers over a network. For example, a file server, a print server, or a mail server.

SET. Secure Electronic Transaction. SET is a protocol that ensures secure transmission of sensitive information, such as data about shoppers, orders, and payment methods.

SGID. Set Group ID.

SHLAP. Shared Library Assistant Process.

Shopper. In Commerce Service Provider, an individual who accesses an electronic store to browse catalogs and make purchases.

Shopper group. A collection of shoppers, as defined by the merchant, who share some common characteristic or shopping pattern. Members of the group may be offered special prices or have customized product information or page designs displayed to them.

Shopping cart. In Commerce Service Provider, a collection of goods or services that a shopper selects from an electronic catalog.

Site. In Commerce Service Provider, a virtual location defined by the existence of a single instance of the Commerce Service Provider product.

SID. Segment ID.

Site administrator. In Commerce Service Provider, an individual with the authority to update the information that is associated with a site.

SIT. Simple Internet Transition.

Site Manager. A collection of data entry forms used to populate the Commerce Service Provider database with mall and site information and to maintain this information. Site Manager is a component of Commerce Service Provider Administrator.

SKIP. Simple Key Management for IP.

SKU number. Stockkeeping Unit number. An alphanumeric identifier for each item of merchandise. It can include variables for department, class, vendor, style, color, size, and location.

SLIH. Second Level Interrupt Handler.

SM. Session Management.

SMIT. System Management Interface Tool.

SMB. Server Message Block.

SMP. Symmetric Multiprocessor.

SNG. Secured Network Gateway.

SP. Service Processor.

SPCN. System Power Control Network.

SPI. Security Parameter Index.

SPM. System Performance Measurement.

SPOT. Shared Product Object Tree.

SQL. (Structured Query Language) A computer language that is used to manipulate relational databases.

SRC. System Resource Controller.

SRN. Service Request Number.

SSA. Serial Storage Architecture.

SSL. Secure Sockets Layer. SSL allows the client to authenticate the server and all data and requests to be encrypted. The URL of a secure server that is protected by SSL begins with https (rather than http). See also authentication.

Staging server. A server on which you can test your data before making it available to shoppers on the production server.

Store. A set of Web pages and macros that work with Commerce Service Provider to allow shoppers to purchase goods and services over the Web. A store may be part of a mall or may exist independently of any other on-line store.

Store administrator. An individual with the authority to update the information that is associated with a store.

Store Creator. A component of the Commerce Service Provider Administrator that allows a site administrator to generate the infrastructure of a new store. It consists of a series of panels that guide the user through the steps required to create a basic store.

Store Manager. A collection of data entry forms used to populate the Commerce Service Provider database with store information and to maintain this information. Store Manager is a component of Commerce Service Provider Administrator.

Store model. One of several templates within the Store Creator that a site administrator can select when designing a new store. The store models present examples of on-line stores that meet the needs of different types of merchants.

STP. Shielded Twisted Pair.

SUID. Set User ID.

SVC. Supervisor or System Call.

SYNC. Synchronization.

TCE. Translate Control Entry.

TCP/IP. Transmission Control Protocol/Internet Protocol. A set of communication protocols that supports peer-to-peer connectivity functions for both local and wide area networks.

Template Designer. A graphical HTML editor used to create and edit templates and HTML files that will be used as Web pages for on-line malls and stores. The Template Designer is a component of Commerce Service Provider Administrator.

TOS. Type Of Service.

TTL. Time To Live.

UCS. Universal Coded Character Set.

UIL. User Interface Language.

ULS. Universal Language Support.

UP. Uni-Processor.

USLA. User-Space Loader Assistant.

UTF. UCS Transformation Format.

UTM. Uniform Transfer Model.

UTP. Unshielded Twisted Pair.

VeriSign. A certifying authority (trusted third party) that can provide a secure certificate.

VFB. Virtual Frame Buffer.

VG. Volume Group.

VGDA. Volume Group Descriptor Area.

VGSA. Volume Group Status Area.

VHDCI. Very High Density Cable Interconnect.

VMM. Virtual Memory Manager.

VP. Virtual Processor.

VPD. Vital Product Data.

VPN. Virtual Private Network.

VSM. Visual System Manager.

WLM. Workload Manage.

XCOFF. Extended Common Object File Format.

XIE. X Image Extension.

XIM. X Input Method.

XKB. X Keyboard Extension.

XOM. X Output Method.

XPM. X Pixmap.

XVFB. X Virtual Frame Buffer.

Index

Symbols

A

B

C

D

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at **ibm.com**/redbooks
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Document Number **Redbook Title**	SG24-5958-00 IBM WebSphere Commerce Suite SPE Customization
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. **ibm.com**/privacy/yourprivacy/

IBM WebSphere Commerce Suite SPE Customization

IBM WebSphere Commerce Suite SPE Customization

Ready-to-use examples for most common customizations

E-commerce solutions for businesses of all sizes

Performance tips for system administrators

IBM Websphere Commerce Suite, Service Provider Edition V3.2, is a complete, scalable, and extendable solution for operating hosted e-commerce services and communities of all sizes. Constructed on the robust Net.Commerce V3.2 software, everything the service provider requires to launch and run a comprehensive hosted e-commerce service is included.

This IBM Redbook contains hints and tips for customizing IBM WebSphere Commerce Suite, Service Provider Edition (WCS SPE) V3.2. These installation tips will help in improving performance and reliability for WCS. This book also demonstrates the flexibility of the product, offers ideas for customization, and gives ready-to-use working examples. How to set up a unique URL for a merchant or customer groups, how to set up cross-selling, and how to customize shipping are also described.

The descriptions provided on installation procedures and common customization give detailed, ready-to-use examples that supply Internet application service providers with the means to successfully customize IBM Websphere Commerce Suite, Service Provider Edition V3.2 and to differentiate their services on the World Wide Web.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by IBM's International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-5958-00 ISBN 0738416258