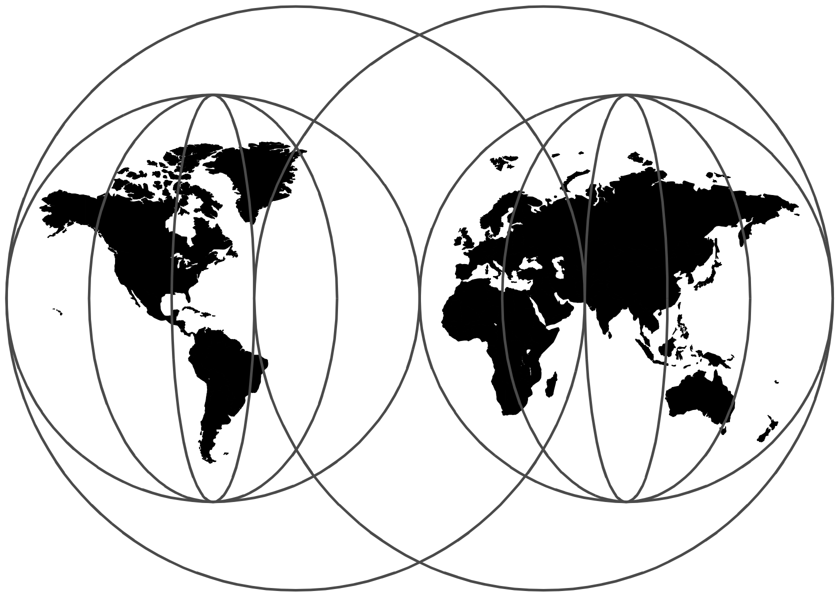
IBM

Customizing Net.Commerce Hosting Server

Trond Norderhaug, Van W. Landrum, Carmen Beavers, Lilien Lam, Daesung Chung



International Technical Support Organization

http://www.redbooks.ibm.com

REDP0022

REDP0022

International Technical Support Organization

Customizing Net.Commerce Hosting Server

November 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix H, “Special Notices” on page 207.

First Edition (November 1999)

This edition applies to 3.12.2 of IBM Net.Commerce Hosting Server for use with the IBM AIX 4.3.2.

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

Contents

Figures

Tables

Preface

This paper gives hints and tips to customize Net.Commerce Hosting Server. The customization scenarios discussed in the paper are based on real world requirements. The Plug-in tool newly introduced in Net.Commerce Hosting Server gives users more flexibility to customize Net.Commerce Hosting Server upon their requirements.

XML and MPG(MultiPurpose code Generation language) are the two essential skills in understanding the ideas presented in this paper. XML is an emerging standard in e-commerce and there are a lot of technical materials to refer to. We included some useful URLs in 2.1.1, “Introduction to XML” on page 13. Information about MPG cab be found in Appendix F, “A MultiPurpose Code Generation language” on page 159. However, MPG is internally used in Net.Commerce Hosting Server. Please note that IBM does not guarantee to provide any technical support for MPG.

This paper consists of three chapters. The first chapter gives an introduction to the Plug-in tool as well as tips to install it on AIX. The second chapter gives an overview of how XML is used in NCHS. The third chapter gives various tips to customize NCHS.

The Team That Wrote This Redbook

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

Trond Norderhaug is a advisory IT specialist in Norway. He/she has 10 years of experience in system integration/application development. His areas of expertise include computer telephony integration(Corepoint), databases(Oracle, DB2, Interbase), application development (C++, Delphi, VB, Perl, Java).

Van W. Landrum is a Technical Support Specialist in the Dallas Systems Center supporting Net.Commerce for IBM in the Americas Advance Technical Support (ATS) organization. He wrote the SmoothStart for Net.Commerce V3 on NT as well as the updated SmoothStart enhancements for IBM Global Services. He has 8 years of experience in the Internet and holds a degree in Business Computer Information Systems from the The University of North Texas. His areas of expertise include eCommerce, Net.Commerce, and relational databases. Before joining ATS, he was the Business Manager for IBM’s *Personal Systems Magazine* and */AIXtra Magazine*.

Carmen Beavers is a Net.Commerce/Internet Developer in Frederick, Maryland, USA. She has 5 years of experience in the Internet Development field. She holds a degree in Computer Science from Virginia Polytechnic Institute and State University. She has worked at Aureus Solutions, Inc., (an IBM Business Partner), for one year. Her areas of expertise include Net.Commerce and web development. She has written primarily on customizing Net.Commerce Hosting Server using Net.Data macros.

Lilien Lam is an e-Business Solutions Specialist in IBM Malaysia. She has 2 years of experience in the e-Commerce field. She holds a Bachelor of Computing and Mathematical Sciences degree from the University of Western Australia, Australia. Her areas of expertise include payment suite products and commerce application

Daesung Chung is working at ITSO, Austin Center, and is in charge of e-business solutions on RS/6000. He writes extensively and teaches IBM classes worldwide on e-business and AIX. He has nine years of experience in AIX, HACMP, and parallel databases on SP, and he has been involved in numerous RS/6000 and SP benchmark cases. Before joining ITSO, he had worked as a senior IT specialist at IBM Korea.

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

Kathy Cacciatore
IBM Austin

Chris Mann
IBM Net.Commerce Hosting Server product manager

Tony Lindsay
IBM Toronto Lab

Mike Polan
IBM Toronto Lab

Quoc Tran
IBM Toronto Lab

John Owczarzak
IBM ITSO Austin Center

Temi Rose
IBM ITSO Austin Center

Steve Gardner
IBM ITSO 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 “ITSO Redpaper Evaluation” on page 215 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. The NCHS Plug-In

1.1 Overview

The Plug-In is more than just a fixpak. It contains a number of enhancement to NCHS. In this section we will give an overview of the new features with reservations to the fact that the programming was not complete and the list of features was still subject to change, when this text was written.

- **Catalog editor**. The catalog editor has been completely rewritten. The new version of the catalog editor is HTML based, as opposed to the old Java based catalog editor. Hence, its not required to have the Java merchant kit installed with your browser to use the catalog editor. However, the page editor is still the same and requires the Java merchant kit to be installed.

  The new HTML based catalog editor runs directly off the Net.Commerce products and categories database tables. This has a number of benefits. As the catalog editor updates the database tables directly, the intermediate step of saving the product information in the file catalog.cdb in a merchants source directory is no longer necessary. Thus, the publish function is no longer needed for products to come on-line. Every change made in the catalog editor is reflected in the on-line store immediately.

  The old publishing step would delete all of the existing products from the database tables and then recreate them from the contents of the catalog.cdb file. This presents a problem when linking from remote Web pages to a specific product using URL’s, because the product reference number in the database will change with every publish. This is not the case with the new catalog editor, since it runs directly off the database tables. The product reference number for a specific product will never change, unless the product is deleted. This creates a new opportunity for the merchant to use the catalog, shopping basket and order management features of NCHS remotely. More details can be found in section 1.2, “New merchant tool features” on page 2.

  Another benefit of the catalog editor is the speed. Since its based on HTML, the time it takes for the catalog editor to start is much shorter.

- **Import catalog data**. It will be possible to use the Net.Commerce Mass Import utility, since the intermediary file called catalog.cdb is no longer used. The CSP can then use the Mass Import utility to load product and category data into the database.

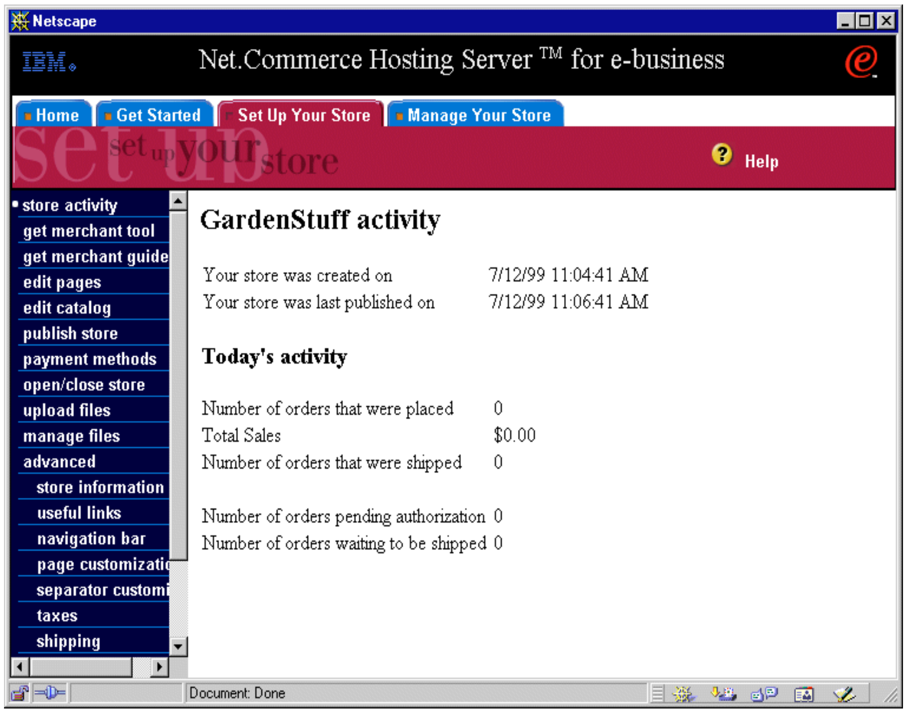
- **Open payments**. The payment wizard and order processing tools will be changed to support Payment Server cassettes. This will allow the CSP to extend NCHS to support new types of payments.

  The order manage tool has been rewritten to support the new payment system, to ease customization and to allow merchants to handle multiple orders with one operation. See the next section for more details.

- **Back-end**. There has been some changes to the NCHS back-end system itself. The main reason is to begin the synchronization of NCHS with the next generation of Net.Commerce. Most notably are the directory changes as a lot of the files previously located in the /usr/lpp/NetCommerce3/CHS are now in /usr/lpp/NetCommerce3/Tools. There will probably be other changes, but since this text is written prior to the release of the Plug-In we don't know yet.

1.2 New merchant tool features

There are a lot of changes in the merchant tool of Plug-In. The figure of new merchant tool is following:



GardenStuff activity

Your store was created on	7/12/99 11:04:41 AM
Your store was last published on	7/12/99 11:06:41 AM

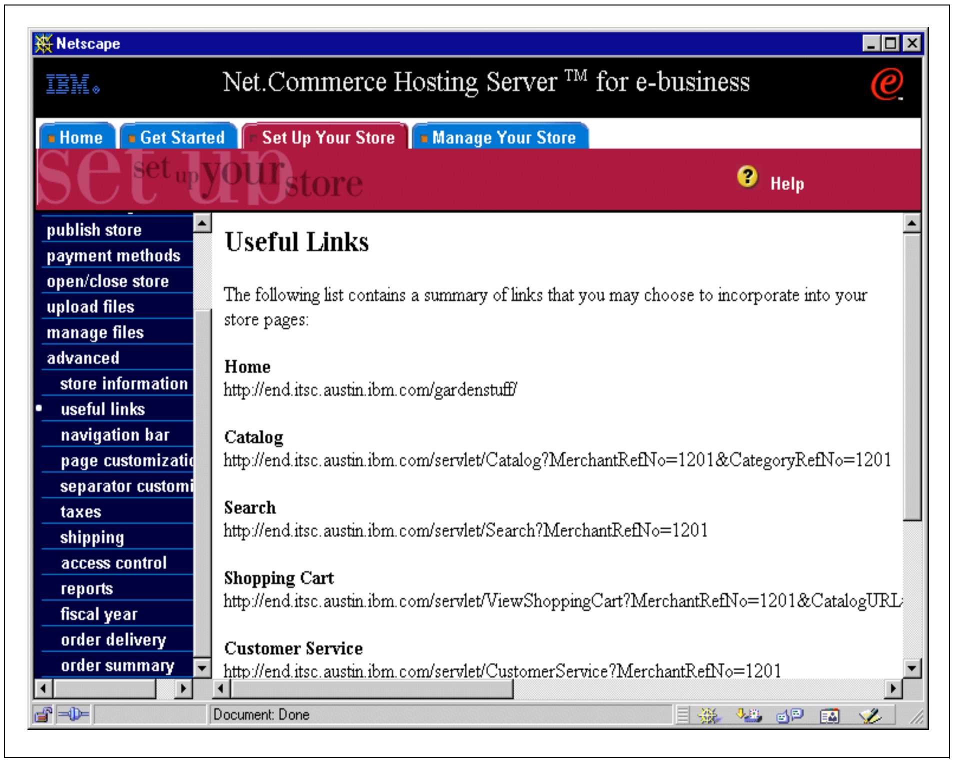
Today's activity

Number of orders that were placed	0
Total Sales	$0.00
Number of orders that were shipped	0
Number of orders pending authorization	0
Number of orders waiting to be shipped	0

Figure 1. Merchant tool of Plug-In

In the **Set Up Your Store** tab, the new catalog editor and some new menus are inserted as you can see in Figure 1. The **page customization** menu and the **separator customization** menu are inserted independent menu items instead of part of **edit pages** menu.

And the **useful links** menu and the **navigation bar** menu are newly inserted. The **useful links** menu provides a summary of links to NCHS static pages. The links include URLs of Home, Catalog, Search, Shopping Cart, Customer Service, Registration, Logon and Logoff pages. By inserting those URLs in the Web page, the merchant can add links from external Web pages to NCHS static pages. In the **navigation bar** menu, you can change URLs of the NCHS static pages. The followings are figures of the **useful links** and the **navigation bar** menu items:



Useful Links

The following list contains a summary of links that you may choose to incorporate into your store pages:

Home
http://end.itsc.austin.ibm.com/gardenstuff/

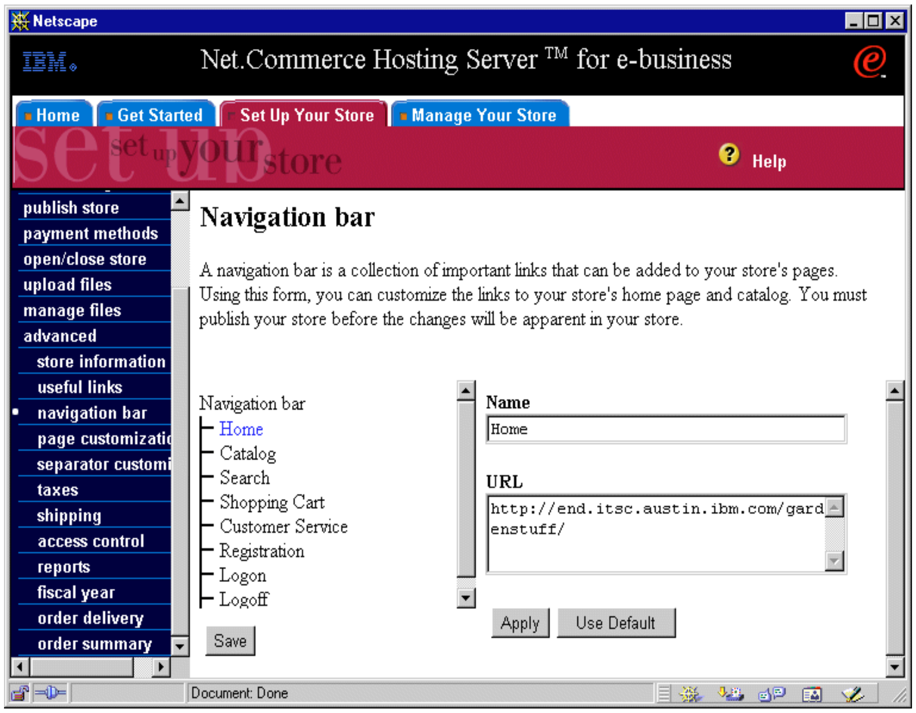
Catalog
http://end.itsc.austin.ibm.com/servlet/Catalog?MerchantRefNo=1201&CategoryRefNo=1201

Search
http://end.itsc.austin.ibm.com/servlet/Search?MerchantRefNo=1201

Shopping Cart
http://end.itsc.austin.ibm.com/servlet/ViewShoppingCart?MerchantRefNo=1201&CatalogURL=

Customer Service
http://end.itsc.austin.ibm.com/servlet/CustomerService?MerchantRefNo=1201

*Figure 2. **useful links** menu of new merchant tool*



Navigation bar

A navigation bar is a collection of important links that can be added to your store's pages. Using this form, you can customize the links to your store's home page and catalog. You must publish your store before the changes will be apparent in your store.

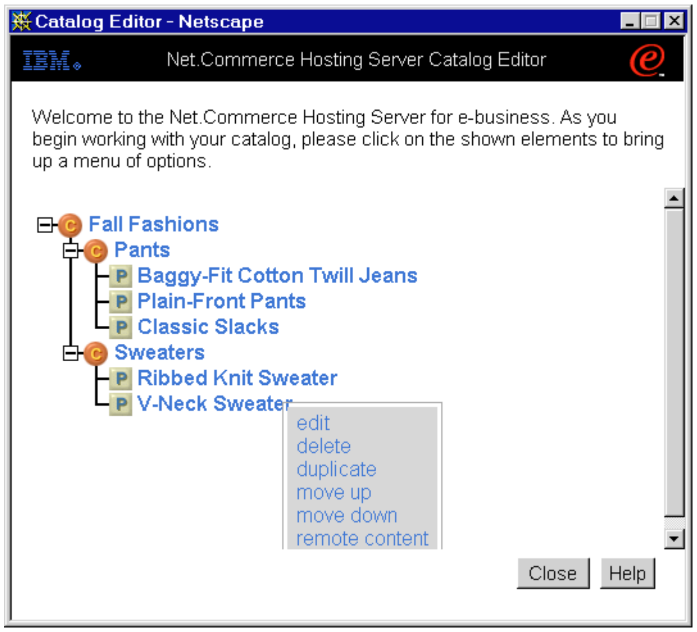
Navigation bar
- Home
- Catalog
- Search
- Shopping Cart
- Customer Service
- Registration
- Logon
- Logoff

Name: Home

URL: http://end.itsc.austin.ibm.com/gardenstuff/

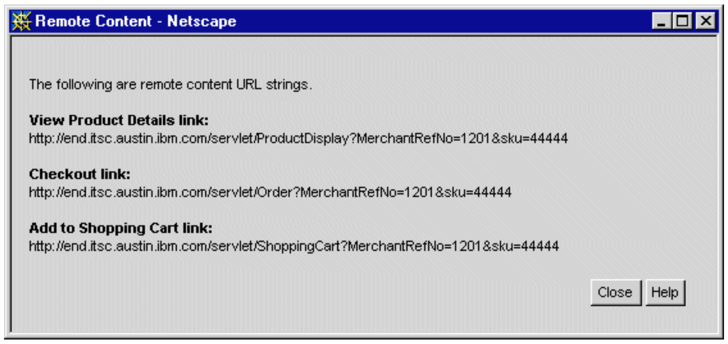
*Figure 3. **navigation bar** menu of new merchant tool*

The new feature introduced in catalog editor is the remote contents of a product. You can see the menu bar about a product item as like in Figure 4 on page 6 if you click left mouse button on the product item. As you can see Figure 5 on page 6, the remote contents of a product displays some useful URLs including View Product Details link, Checkout link and Add to Shopping Cart link. The merchant can add link to display a product details page, link to order a product and link to add a product to shopping cart by inserting those URLs in the Web pages.



Welcome to the Net.Commerce Hosting Server for e-business. As you begin working with your catalog, please click on the shown elements to bring up a menu of options.

Figure 4. New catalog editor



The following are remote content URL strings.

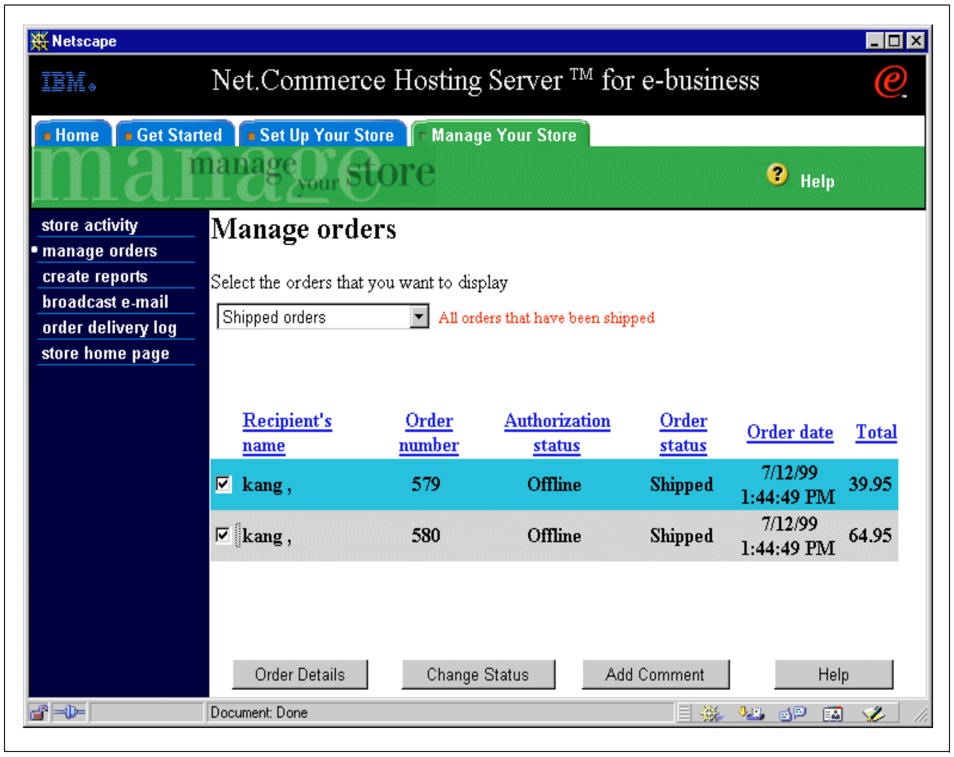
View Product Details link:
http://end.itsc.austin.ibm.com/servlet/ProductDisplay?MerchantRefNo=1201&sku=44444

Checkout link:
http://end.itsc.austin.ibm.com/servlet/Order?MerchantRefNo=1201&sku=44444

Add to Shopping Cart link:
http://end.itsc.austin.ibm.com/servlet/ShoppingCart?MerchantRefNo=1201&sku=44444

Figure 5. Remote content of a product

In the **manage orders** menu of **Manage Your Store** tab, multiple selection feature is newly introduced. You can select more than one order by checking the check box in the front of the order list. So the merchant can change status of multiple orders and add comment to multiple orders at once. The figure of the **manage orders** menu is as like below:



Manage orders

Select the orders that you want to display

Shipped orders — All orders that have been shipped

	Recipient's name	Order number	Authorization status	Order status	Order date	Total
☑	kang ,	579	Offline	Shipped	7/12/99 1:44:49 PM	39.95
☑	kang ,	580	Offline	Shipped	7/12/99 1:44:49 PM	64.95

Figure 6. ***manage orders*** *menu of new merchant tool*

1.3 Installing the Plug-in on an existing NCHS system

Before you can install Net.Commerce Hosting Server V3.1.2 Plug-in (referred to as the plug-in), you must have already installed Net.Commerce Hosting Server V3.1.2.2. Otherwise the Plug-in will not work properly.

The installation of the plug-in consists of extracting a compressed file and configuration modifications. If you have existing information from an installation of Net.Commerce Hosting Server V3.1.2, you need to perform a migration procedure after installing the plug-in.

To prepare to install the plug-in, do the following:

- Stop the Net.Commerce, Websphere and the Web server services.
- If you are migrating information, backup your system.

- Download the plugin binary NCHSPluginAIX.tar from `http://www.software.ibm.com/commerce/net.commerce/community/hosthood/downloads/`. The tar file contains three files.

```
# tar -tvfNCHSPluginAIX.tar
-r-xr-xr-x   2 2   648382 Aug 30 19:07:41 1999 NCHSPlugin.pdf
-r-xr-xr-x   2 2  9573971 Aug 31 15:18:45 1999 NCHSPlugin.tar.Z
-r-xr-xr-x   2 2     3933 Aug 29 19:07:26 1999 readme_plugin.txt
```

Follow the below steps to install Plug-in.

1. Extract the plug-in file (NCHSPlugin.tar.Z) and uncompress it.

```
# tar -xvfNCHSPluginAIX.tar NCHSPlugin.tar.Z
# uncompress NCHSPlugin.tar
```

2. Untar the NCHSPlugin.tar into /usr/lpp/NetCommerce3

```
# cd /usr/lpp/NetCommerce3
# tar -xvf/tmp/NCHSPlugin.tar
```

This will install the updated files into the appropriate directories.

3. Open the jvm.properties file. This file is located in the following directory:

/usr/lpp/IBMWebAS/properties/server/servlet/servletservice

To the beginning of the ncf.jvm.classpath value, add the following (on one line):

```
/usr/lpp/NetCommerce3/Tools/lib/nctools.jar:
/usr/lpp/NetCommerce3/Tools/lib/nchs.jar:
/usr/lpp/NetCommerce3/Tools/:
/usr/lpp/NetCommerce3/Tools/lib/xml4j.jar:
```

Save the file.

4. Update the Web server configuration file by following these steps:

Open /etc/httpd.conf

Then add the following statement:

```
Pass /NCTools/* /usr/lpp/NetCommerce3/Tools/public/*
It should be placed before
Pass          /*            /usr/lpp/internet/server_root/pub/*
```

Save the file.

1.4 Migrating on AIX

In the following procedure, we assume you are using Domino Go webserver.

To run the migration procedure do the following:

1. You can begin the migration procedure only after you have completed steps 1 through 4 of Installing the Net.Commerce Hosting Server V3.1.2 Plug-in.
2. Logon as the DB2INSTANCE owner ID.
3. Ensure that you have the Java Development Kit, version 1.1.6 installed.

```
$ lslpp -l Java*
  Fileset
-----------------------------------------------------------------------
Path: /usr/lib/objrepos
  Java.adt.docs            1.1.6.0  COMMITTED  Java Documentation
  Java.adt.includes        1.1.6.0  COMMITTED  Java Application
                                               Development Toolkit
  Java.adt.src             1.1.6.1  COMMITTED  Java Class Source Code
  Java.rte.Dt             1.1.6.0  COMMITTED  Java Runtime Environment
                                               Desktop
  Java.rte.bin            1.1.6.1  COMMITTED  Java Runtime Environment
                                               Executables
  Java.rte.classes        1.1.6.1  COMMITTED  Java Runtime Environment
                                               Classes
  Java.rte.lib            1.1.6.1  COMMITTED  Java Runtime Environment
                                               Libraries
```

4. Edit $HOME/.profile and change $CLASSPATH variable to include the following information:

```
<DB2_instance_directory>/sqllib/java/db2java.zip
/usr/lpp/NetCommerce3/Tools/lib/xml4j.jar
/usr/lpp/NetCommerce3/Tools/lib/nctools.jar
/usr/lpp/NetCommerce3/Tools/lib/nchs.jar
/usr/lpp/NetCommerce3/Tools/
/usr/lpp/NetCommerce3/CHS
/usr/lpp/NetCommerce3/CHS/CHS.jar
```

   where <DB2_instance_directory> is the directory in which DB2INSTANCE is installed.

5. Start Net.Commerce, Websphere and the Web server.
6. Add the following text to LIBPATH (ensure that you are logged on as the DB2INSTANCE owner ID).

   /usr/lpp/NetCommerce3/bin

Do the following.

```
$ . $HOME/.profile
```

7. Do the following to look into which stores were created.

```
$ db2 select mpdirname,mpmenbr,mpmetaloc from mcspinfo
```

```
1            MPMENBR     3
------------ ----------- ----------------------------------------
chssamplestore  401 /usr/lpp/NetCommerce3/CHS/source/401
chsservicesstore  402 /usr/lpp/NetCommerce3/CHS/source/402
gardenstuff  1478 /usr/lpp/NetCommerce3/CHS/source/1478
```

Publish existing stores by typing the following command at a command prompt (on one line):

```
$ java com.ibm.commerce.tools.nchs.migration.Migration ncadmin ncadmin
/usr/lpp/NetCommerce3/Tools/config/config.xml
```

```
$ java com.ibm.commerce.tools.nchs.migration.Migration ncadmin ncadmin
/usr/lpp/NetCommerce3/Tools/config/config.xml
1999.09.03 12:45:15.610 [main] DEBUG - Parsing XML file:
/usr/lpp/NetCommerce3/Tools/config/config.xml.
Login as administrator..
Begin migration step
Succeeded in publishing merchant 401 .
Succeeded in publishing merchant 402 .
Succeeded in publishing merchant 1478 .
End migration step
```

8. Open the following file:

/usr/lpp/IBMWebAS/properties/server/servlet/servletservice/servlets.properties

Comment out the following lines:

```
servlet.MerchantAdmin.code=com.ibm.chs.common.PageManager
servlet.MerchantAdmin.initArgs=configfile=/usr/lpp/internet/server_root/pu
b/ncommerce.conf
```

Add the following to the end of the file:

```
# Request Manager servlet
servlet.MerchantAdmin.code=com.ibm.commerce.tools.request_management.Reque
stManager
servlet.MerchantAdmin.initArgs=configfile=/usr/lpp/NetCommerce3/Tools/conf
ig/config.xml
servlet.ProductDisplay.code=com.ibm.commerce.tools.nchs.urlProxy.ProductDi
splayProxy
```

```
servlet.ShoppingCart.code=com.ibm.commerce.tools.nchs.urlProxy.ShoppingCar
tProxy
servlet.Order.code=com.ibm.commerce.tools.nchs.urlProxy.OrderProxy
servlet.Register.code=com.ibm.commerce.tools.nchs.urlProxy.RegisterProxy
servlet.Logon.code=com.ibm.commerce.tools.nchs.urlProxy.LogonProxy
servlet.CustomerService.code=com.ibm.commerce.tools.nchs.urlProxy.Customer
ServiceProxy
servlet.ViewShoppingCart.code=com.ibm.commerce.tools.nchs.urlProxy.ViewSho
ppingCartProxy
servlet.Search.code=com.ibm.commerce.tools.nchs.urlProxy.SearchProxy
servlet.Home.code=com.ibm.commerce.tools.nchs.urlProxy.HomeProxy
servlet.Logoff.code=com.ibm.commerce.tools.nchs.urlProxy.LogoffProxy
servlet.Catalog.code=com.ibm.commerce.tools.nchs.urlProxy.CatalogProxy
# Custom servlets
servlet.ExtendedPaymentFinish.code=custom.nchs.payment.ExtendedPaymentFini
sh
servlet.ExtendedChangeStatus.code=custom.nchs.order_mgmt.ExtendedChangeSta
tus
servlet.ExtendedAddComments.code=custom.nchs.order_mgmt.ExtendedAddComment
s
servlet.GetAuthorizationStatus.code=custom.nchs.order_mgmt.GetAuthorizatio
nStatus
servlet.ExtendedPaymentInitialize.code=custom.nchs.payment.ExtendedPayment
Initialize
```

9. Stop and restart Net.Commerce, Websphere and the Web server.

10. Ensure that you are logged on as the DB2INSTANCE owner id and publish stores using the plug-in method by typing the following command at a command prompt (on one line):

    ```
    $ java com.ibm.commerce.tools.nchs.migration.PostMigration ncadmin
    ncadmin /usr/lpp/NetCommerce3/Tools/config/config.xml
    ```

    If errors occurred during the migration process, they are logged in /usr/lpp/NetCommerce3/Tools/logs/tools.log

```
$ java com.ibm.commerce.tools.nchs.migration.PostMigration ncadmin \
ncadmin /usr/lpp/NetCommerce3/Tools/config/config.xml
1999.09.03 12:48:15.501 [main] DEBUG - Parsing XML file: \
/usr/lpp/NetCommerce3/Tools/config/config.xml.
Login as administrator..
Begin migration step
Begin updating PRODUCT table: change directory path of  uploaded product image f
iles
End updating PRODUCT table
Succeeded in  creating setting file for merchant 1478 .
Succeeded in publishing merchant 1478 .
Succeeded in  creating setting file for merchant 401 .
Succeeded in publishing merchant 401 .
Succeeded in  creating setting file for merchant 402 .
Succeeded in publishing merchant 402 .
End migration step
```

11. Stop and restart Net.Commerce, Websphere and the Web server.

Chapter 2. Overview of NCHS customization

2.1 Overview XML in NCHS Plugin

2.1.1 Introduction to XML

XML was introduced into NCHS by the Plug-in. Most of the configuration files in NCHS are now in XML format. If you are unfamiliar with XML, here is a brief introduction.

XML is a method for putting structured data in a text file

For “structured data" think of such things as spreadsheets, address books, configuration parameters, financial transactions, technical drawings, etc. Programs that produce such data often also store it on disk, for which they can use either a binary format or a text format. The latter allows you, if necessary, to look at the data without the program that produced it. XML is a set of rules, guidelines, conventions, whatever you want to call them, for designing text formats for such data, in a way that produces files that are easy to generate and read (by a computer), that are unambiguous, and that avoid common pitfalls, such as lack of extensibility, lack of support for internationalization/localization, and platform-dependency.

XML looks a bit like HTML but isn’t HTML

Like HTML, XML makes use of tags (words bracketed by '<' and '>') and attributes (of the form name="value"), but while HTML specifies what each tag & attribute means (and often how the text between them will look in a browser), XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. In other words, if you see "<p>" in an XML file, don't assume it is a paragraph. Depending on the context, it may be a price, a parameter, a person, a p... (b.t.w., who says it has to be a word with a "p"?)

XML is text, but isn’t meant to be read

XML files are text files, but not meant to be read by humans. They are text files, because that allows experts (such as programmers) to more easily debug applications, and in emergencies, they can use a simple text editor to fix a broken XML file. But the rules for XML files are much stricter than for HTML. A forgotten tag, or a an attribute without quotes makes the file unusable, while in HTML such practice is often explicitly allowed, or at least tolerated. It is written in the official XML specification: applications are not allowed to try to second-guess the creator of a broken XML file; if the file is broken, an application has to stop right there and issue an error.

Simple example of XML Usage

The best way to appreciate what XML documents look like is with a simple example. Imagine your company sells products on-line. Marketing descriptions of the products are written in HTML, but names and addresses of customers, and also prices and discounts are formatted with XML. Here is the information describing a customer:

```
<customer-details id="AcPharm39156">
        <name>Acme Pharmaceuticals Co.</name>
        <address country="US">
            <street>7301 Smokey Boulevard</street>
            <city>Smallville</city>
            <state>Indiana</state>
            <postal>94571</postal>
        </address>
    </customer-details>
```

The XML syntax uses matching start and end tags, such as <name> and </name>, to mark up information. A piece of information marked by the presence of tags is called an element; elements may be further enriched by attaching name-value pairs (for example, country="US" in the example above) called attributes. Its simple syntax is easy to process by machine, and has the attraction of remaining understandable to humans. XML is based on SGML, and is familiar in look and feel to those accustomed to HTML.

Setting XML into C++ context

We can view an XML element as a class and attributes as properties of a class. This allows us to relate the information in the previous example as follows :

```
customer-details.id
customer-details->name
customer-details->address.country
customer-details->address->street
customer-details->address->state
```

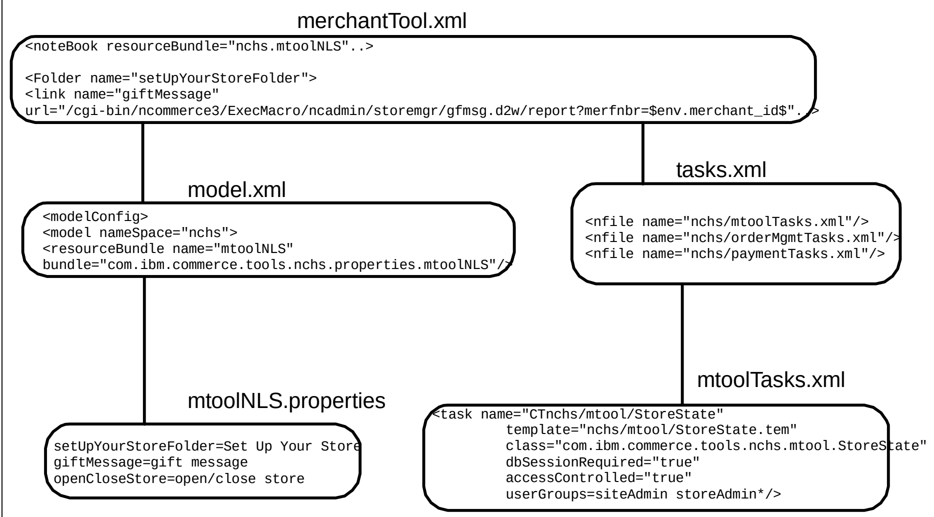
For further reference, check the following sites.

http://www.xmlbooks.com/

http://www.ibm.com/developerWorks

2.1.2 XML configuration files in NCHS

The following diagram shows the relationship among NCHS configuration files.



```
<noteBook resourceBundle="nchs.mtoolNLS"..>

<Folder name="setUpYourStoreFolder">
<link name="giftMessage"
url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/gfmsg.d2w/report?merfnbr=$env.merchant_id$"../>
```

```
<modelConfig>
<model nameSpace="nchs">
<resourceBundle name="mtoolNLS"
bundle="com.ibm.commerce.tools.nchs.properties.mtoolNLS"/>
```

```
<nfile name="nchs/mtoolTasks.xml"/>
<nfile name="nchs/orderMgmtTasks.xml"/>
<nfile name="nchs/paymentTasks.xml"/>
```

```
setUpYourStoreFolder=Set Up Your Store
giftMessage=gift message
openCloseStore=open/close store
```

```
<task name="CTnchs/mtool/StoreState"
        template="nchs/mtool/StoreState.tem"
        class="com.ibm.commerce.tools.nchs.mtool.StoreState"
        dbSessionRequired="true"
        accessControlled="true"
        userGroups=siteAdmin storeAdmin*/>
```

Figure 7. Relationship among xml files

2.2 Impact of customizations

We have examined the impact of modifying some of NCHS configurations. The directory table describes sub directories of standard installation directory (/usr/lpp/NetCommerce3/).

Changes to files that may affect store(s).

directory	impact(s)
/CHS/source/<merchant_id>/	only this store, changes may be erased when publishing changes from MerchantTool
/macro/common/	by default all stores, depends on changes to local macro’s
/macro/<LANG>/<merchant_id>/	local macro’s, default just a link to the common macro.

directory	impact(s)
/Tools/	All merchants are affected by any change to any file. These changes will affect the MerchantTool/Catalog/Payment and all other parts of NCHS Plugin.

Chapter 3. NCHS advanced customization

There are many attractive things that can be customized, but the general documentation on these items is very limited. This chapter will explain some advanced customization features of NCHS and provide examples on how to do the customization. Some of the issues discussed in this chapter require extensive knowledge about the workings of Net.Commerce. The various sections discuss customization of NCHS which requires understanding of Net.Data, SQL, C++, MPG (MultiPurpose Code Generation Language) and Net.Commerce Tasks, Commands and Overridable Functions. The MPG language is specific to NCHS and is described in Appendix F, “A MultiPurpose Code Generation language” on page 159.

3.1 Overview

Customization of NCHS is an advanced topic as the documentation is very limited. You have to be familiar with Net.Commerce and you have to spend time with NCHS to understand how it works. In this chapter we will provide you with some examples of how NCHS can be customized. This should help you to understand the customization issues with NCHS, and let you do your own customization.

The basic idea of NCHS is to let a CSP provide and sell a commerce service to as many merchants as possible. To keep the cost down for both the CSP and the merchants, some common elements are used for all the stores in a hosted commerce environment. Hence, the merchant tool and the store model of NCHS. This is a key issues to remember when customizing NCHS.

The desire to do customization for one particular merchant should cause the big red lights to flash. Is it worth it to do customization for one particular merchant or should that merchant be moved to an independent Net.Commerce instance? If you do customization for one merchant do you then diminish the benefits of large-scale operations that NCHS provides? If the merchant decides not to be a customer of yours anymore, can you then remove the customization without damaging NCHS and the rest of the merchants?

Even if the above is true there may still be reasons to customize NCHS for one particular merchant or for a group of merchants. This is not an easy task with the current version of NCHS (NCHS 3.1.2). It will be easier in future versions of NCHS, but right now you will have to do most of the thinking and development your self. However, it is absolutely possible to do merchant specific customization. We will not provide any examples of how to do it, but you should be able to figure it out from the examples we do provide.

The following text takes a starting point in a new store being created and then explains what happens during the creation and publishing of the store. It's a key issue to understand the publish process before doing any customization as it creates and makes changes to files and database contents.

3.2 The merchant store model

This section will explain what happens when a merchant creates a new store and what happens when the merchant publishes the store.

3.2.1 Creating a new store

When a merchant decides to create a new store a number of things happens within NCHS.

First a new merchant record is created in the database and a merchant reference number (MERFNBR in the MERCHANT table) is assigned to the new merchant.

The unique merchant reference number is used to create a directory to store the HTML pages and catalog data for that particular merchant. The directory is created as /NetCommerce3/CHS/source/*id* where the id is the merchant reference number.

Here is a listing of /NetCommerce3/CHS/source

```
[.]      [..]     [1051]   [1052]   [1951]   [2676]   [2677]   [2678]
[2876]   [3076]   [3376]   [3626]   [null]   [sample]
```

The new directory will be preloaded with a layout file called site.sdb which contains HTML pages and an XML file called settings.xml which contains some default settings for the site.

Here is a listing of /NetCommerce3/CHS/source/1052

```
[.]           [..]          catalog.cdb    [html]         settings.xml
site.sdb
```

The layout file describes all the content that the merchant creates with the edit pages functions. The data in the layout file is not on-line yet and shoppers will therefore not be able to see the contents in the layout file until the store is published. To put the contents of the layout file on-line, the merchant must publish his store using the merchant tool. How the publish function works will be explained in the next section.

When a store is created it is preloaded with a layout file as described above. The preloaded layout file is created from a default layout file. Basically, it is a copy and rename of file located in the directory CHS/layout/en_US.

- CHS/layout/en_US/layout.sdb $\longrightarrow$ CHS/source/*id*/site.sdb

3.2.2 Changing the default store layout

As the logic above shows, all stores are created from the same default layout file. It also means that we can create a new default layout file and make all new stores use them. The procedure for doing that is:

- Create a new store.
- Use the **edit pages** functions of the merchant tool to edit the pages of your store. Save your work, but do NOT publish your store as NCHS will write store specific data to the site.sdb file, and thereby making it useless as a generic file.
- Find the merchant reference number for the new store. There are a number of ways you can do this. One way is to look in the MERCHANT table in the database, as shown below. You have to be logged in as the database instance owner (db2inst1 in our case).

```
$ id
uid=201(db2inst1) gid=201(db2iadm1) groups=202(smadmin)

$ db2 select merfnbr,mestname from merchant

MERFNBR     MESTNAME

----------- --------------------------------------------------------------------

        526 daisy's shop

        527 FlowerShop

       1476 23June

       1477 test

       1478 Garden Stuff

       1851 24June

       1852 donald's shop
```

Pick the merchant reference number next to your store name. Since the store created for the example in this book is called Garden Stuff, the number is 1478.

- You should probably backup the original layout file in CHS/layout/en_US before continuing.
- Replace the default layout file by copying the site.sdb file from your store over the default layout file. In this step you have to use the merchant reference number we found earlier as the id:
  - CHS/source/*id*/site.sdb → CHS/layout/en_US/layout.sdb
- All new shops will now be created with the new default layout file.

3.2.3 The merchants HTML directory

Another directory will also be created inside the newly created merchant reference number directory CHS/source/*id*. This directory is called html and its path is then CHS/source/*id*/html. When first created the html directory contains a few .html files.

```
[.]                 [..]                  About_Us125.html
catalog.html        customer_service.html index.html
logon.html          register.html         search.html
shop_cart.html      zzz.html
```

The files in the html directory will be copied to the merchants on-line directory when the store is published. The merchant can actually access all the files in the html directory, by using the merchant tool function **manage files**. It is also this directory the merchant uploads his own html files with the **upload files** function.

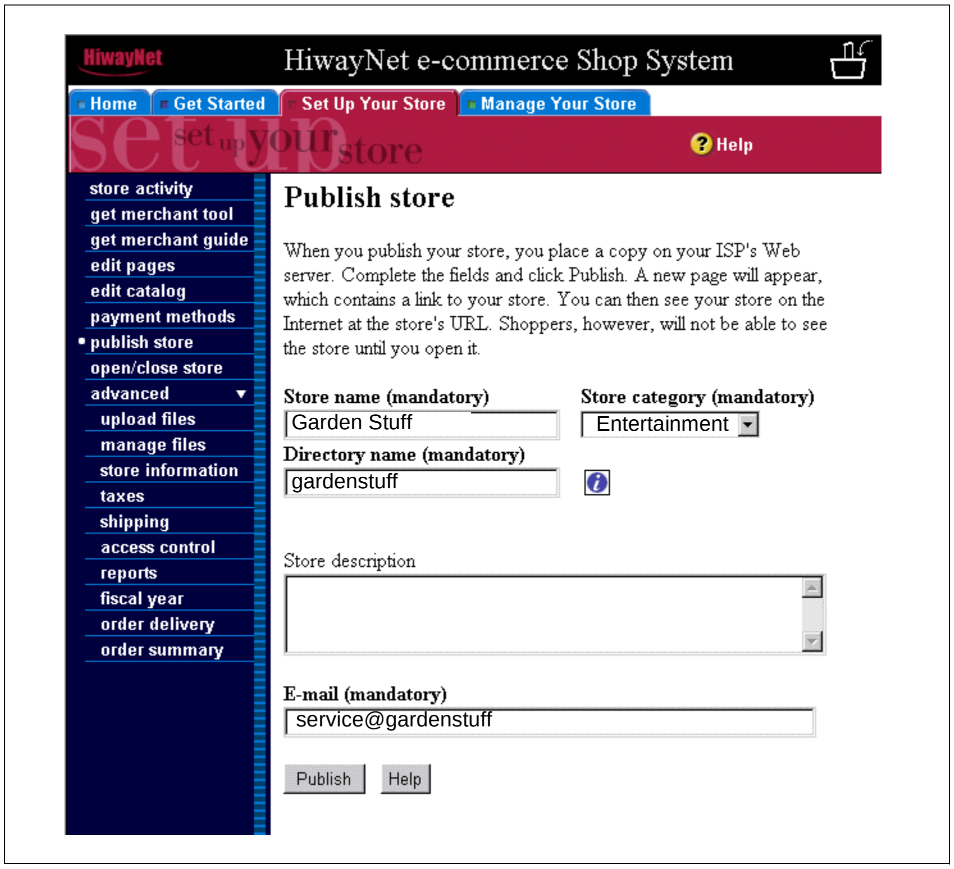
The preloaded html files just contain redirections, so if a shopper for example tries to access http://hostname/storename/catalog then the catalog.html file will be read and the shopper’s browser will be redirected to the catalog pages of the store. The redirection for the catalog consists of the Net.Commerce command CategoryDisplay with the proper parameters. This is how all the above nine preloaded html files work.

The merchant or the CSP can modify or delete the preloaded files, as they will not be recreated by NCHS. The merchant could also upload other files including html files created with non-NCHS authoring tools (for example TopPage, NetObjects Fusion or Frontpage). However, NCHS will create or recreate some html files based on the contents of site.sdb. The default site.sdb contains index.html and an About_Us page, but could contain more pages. Hence, if a merchant uploads a file, for example his own index.html, and publishes his store again, NCHS will overwrite the index.html that was just uploaded. This can be avoided by deleting all the pages in site.sdb using the merchant tool. The procedure on how to prevent the customized html files from being overwritten while the store is published is discussed later.

A number of things are done in the database when a merchant creates a new store, such as adding the merchants user id and password to the SHOPPERS table, creating a record for the store in MCSPINFO and some other things. Going into details about what is put into the database is beyond the scope of this book.

3.2.4 Publishing a store

The **publish** function in the merchant tool is used by a merchant to publish a store.



Publish store

When you publish your store, you place a copy on your ISP's Web server. Complete the fields and click Publish. A new page will appear, which contains a link to your store. You can then see your store on the Internet at the store's URL. Shoppers, however, will not be able to see the store until you open it.

Store name (mandatory): Garden Stuff
Store category (mandatory): Entertainment
Directory name (mandatory): gardenstuff
Store description:
E-mail (mandatory): service@gardenstuff

Figure 8. The publish window

Most of the fields on the Publish store page, as shown on Figure 8, are mandatory. This is because they are used by central functions of NCHS. In this section we will focus on the parameter used by the publish function, and that is the directory name.

The directory name, gardenstuff in our example, is used during the publish process and is also stored in the database (the table is MCSPINFO). The first thing the directory name is used for is to create the directory that will hold the various HTML files for the shop:

- For AIX: /usr/lpp/internet/server_root/pub/*directoryname*
  or
- *For NT: /IBM/www/html/directoryname.*

Since the directory name is created in the root of the html directory it will also become the URL for the shop, as: http://hostname/*directoryname*

MCSPINFO table

```
$ db2 select mpdirname,mpmenbr,mpmetaloc from mcspinfo\
where mpdirname='gardenstuff'

1            MPMENBR     3
------------ ----------- ----------------------------------------
gardenstuff         1478 /usr/lpp/NetCommerce3/CHS/source/1478
```

NCHS checks that the directory name is not already in use by another merchant and that the directory is not contained in the file ReservedDirectoryNames.properties before the merchant can use it. The ReservedDirectoryNames file is located in the CHS/properties directory and contains a list of reserved directory names that merchants can not select. A text editor can be used to add new directories to the ReservedDirectoryNames.properties file if desired.

If a merchant decides to change directory name, then NCHS will clean up the directory with the old name (deleting all the files in the directory including the directory itself). The new directory name will then be used in the publish process.

Other store specific directories are created using the merchant reference number. These directories are as follows:

- macro/en_US/*merchantRefNum*
- macro/en_US/category/*merchantRefNum*
- macro/en_US/product/*merchantRefNum*

These directories contain macros specific to each of the stores. Because they use the merchant reference number instead of the home directory name, they will not change when the home directory name changes.

The Net.Data macro files

By default all the stores in NCHS use the same shopping flow and checkout flow. The macro files that all the stores share are in the directory macro/en_US/CSPstoremodel. However, the merchants get three of their own macro directories, as shown above, because some of the content in a store is customized towards a merchant. This is for example the background color or graphics, the store name or the horizontal separator between products.

This is handled by creating a set of small and simple macro files for each merchant. These macro files simply includes a merchant specific data file and the macro file from the CSPstoremodel directory. The file shown below is a merchants logon.d2w macro file.

```
logon.d2w:

%include "1478/gardenstuff.inc"
%include "CSPstoremodel/logon.d2w"
```

The merchant has selected the directory name gardenstuff and the merchants logon.d2w file is in macro/en_US/gardenstuff. As the file shows, it includes a file called gardenstuff.inc. This file contains all the merchant specific settings, like background color, merchant name and so on. The second file included is the logon.d2w from the NCHS store model directory, and it contains all the SQL and HTML to create a page for the shopper.

During the first publish about 24 marco files will be generated in the merchants macro directory. These macro files will only be generated the first time the merchant uses a directory name. If the merchant change directory name, then all the files will be regenerated for that new directory.

Since the macro files are only generated once, the CSP can change these files in any way. The CSP could for example let some merchants use another store model by simply changing the included macro file to something like this:

```
logon.d2w:

%include "1478/gardenstuff.inc"
%include "MyNewStoreModel/logon.d2w"
```

As long as the merchant does not change their home directory name, then the above will work.

Since the include file (gardenstuff.inc in the example above) is in the directory named by the merchant reference number, the file is not generated every time a merchant executes the publish function, so any changes to the .inc file will not be lost when the publish function is performed.

The last two macro directories, as described in, “The Net.Data macro files” on page 23, are

macro/en_US/category/*id,*
macro/en_US/product/*id*

and they each contain one macro file that looks like the following:

```
%include "1052\include.inc"
%include "CSPstoremodel\csp_cat.d2w
```

3.3 Changing shopping flow

3.3.1 Default Shopping and checkout flow

What is kept after a publish

When a merchant publishes his/her store, html files are generated from the contents in site.sdb and are written to /usr/lpp/NetCommerce3/CHS/source/*id*/html, overwriting all html files with the same names. Hence, changes made to the html files are not kept. However, change made to the macro files in /en_US/*id,* macro/en_US/category/*id, and* macro/en_US/product/*id* are preserved.

As described in 3.2.4, “Publishing a store” on page 21, all the stores in NCHS follow the same shopping flow and checkout flow by default. The shopping and checkout flows are the web pages a shopper moves through when making a purchase in a store. When the shopper selects categories, looks at products and adds them to the shopping cart it is called the shopping flow. When the shopper begins to finalize an order by entering a billing address, a shipping address and the payment information then it is called the checkout flow. Figure 9 on page 26 shows the default shopping and checkout flows of the CSP store model.

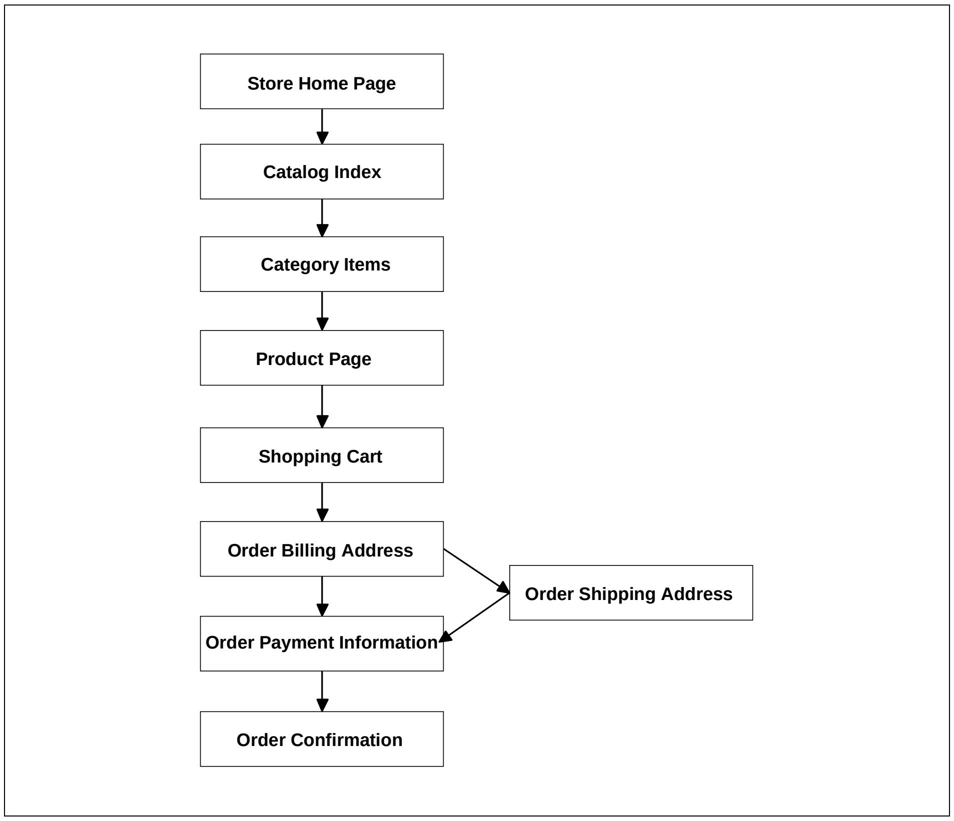
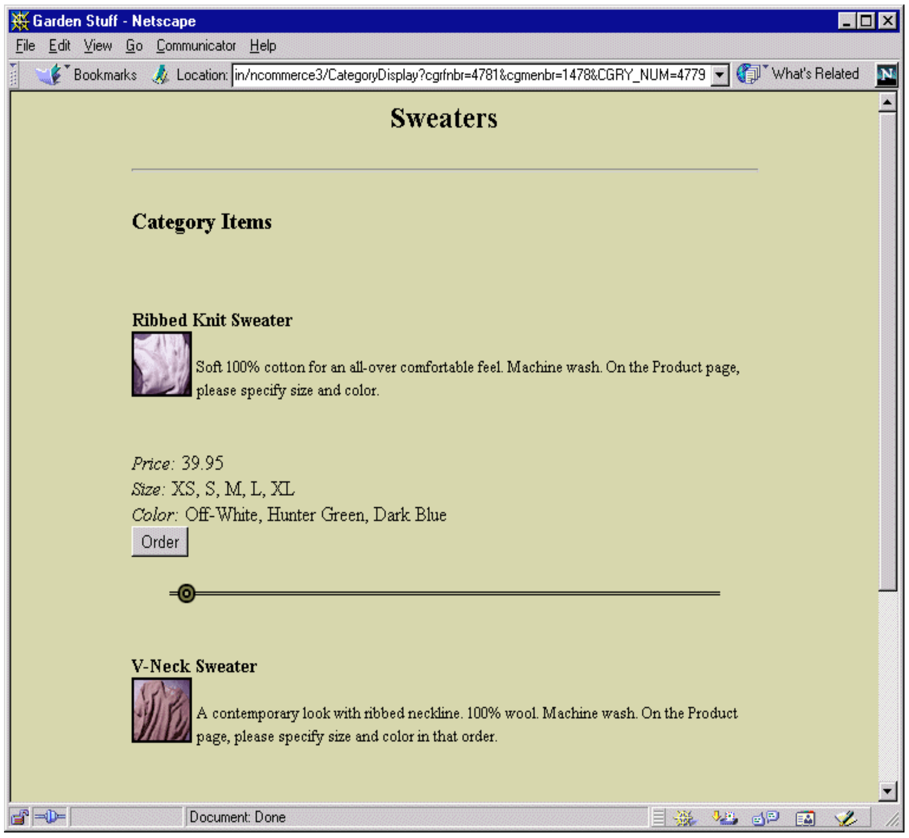


Figure 9. Shopping flow and checkout flow of the CSP store model

A shopper can browse catalog index page by selecting **Catalog** menu in the navigation bar of the store home page.The catalog index page shows product category list of the store. When the shopper selects the category that he/she wants to see, the category items page shows product list of the selected category as shown below:


Garden Stuff - Netscape
File Edit View Go Communicator Help
Bookmarks Location: in/ncommerce3/CategoryDisplay?cgrfnbr=4781&cgmenbr=1478&CGRY_NUM=4779
What's Related

Sweaters

Category Items

Ribbed Knit Sweater

Soft 100% cotton for an all-over comfortable feel. Machine wash. On the Product page, please specify size and color.

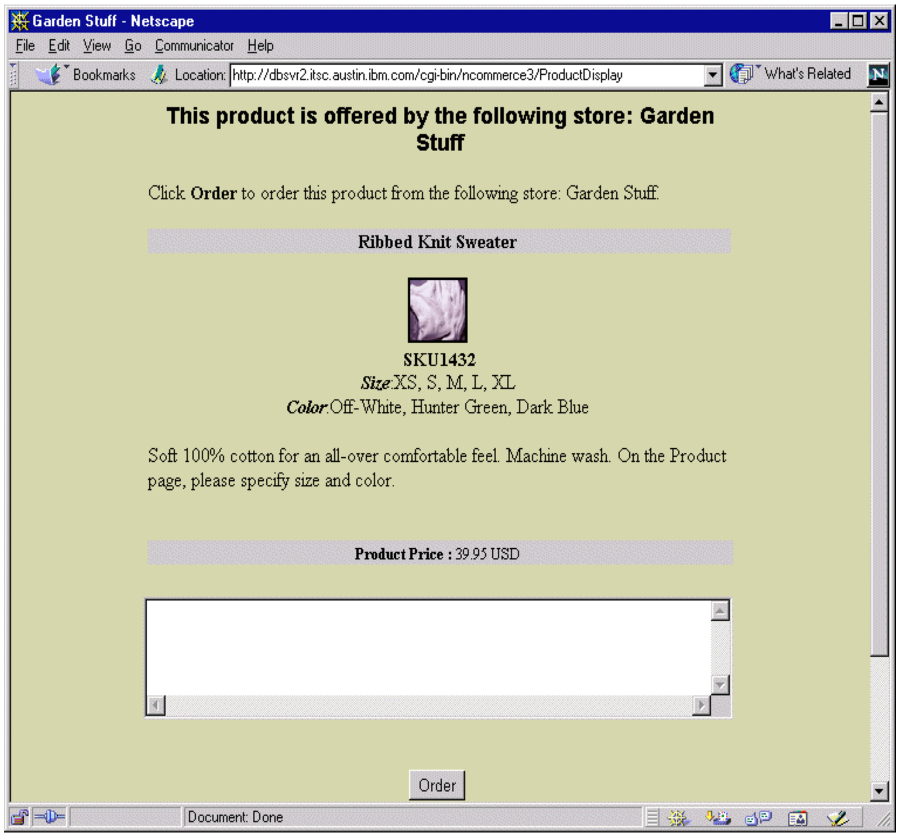
Price: 39.95
Size: XS, S, M, L, XL
Color: Off-White, Hunter Green, Dark Blue

V-Neck Sweater

A contemporary look with ribbed neckline. 100% wool. Machine wash. On the Product page, please specify size and color in that order.

Figure 10. Category items page of the CSP store model

If the shopper clicks the **Order** button below the product description, they can see the product page as like Figure 11 on page 28. A comment can be entered in the product page when a product is ordered so that attributes such as size or color can be specified. You can see the text area for the comment above the **Order** button in the product page.


Garden Stuff - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://dbsvr2.itsc.austin.ibm.com/cgi-bin/ncommerce3/ProductDisplay
What's Related
Order
Document: Done

This product is offered by the following store: Garden Stuff

Click **Order** to order this product from the following store: Garden Stuff.

Ribbed Knit Sweater

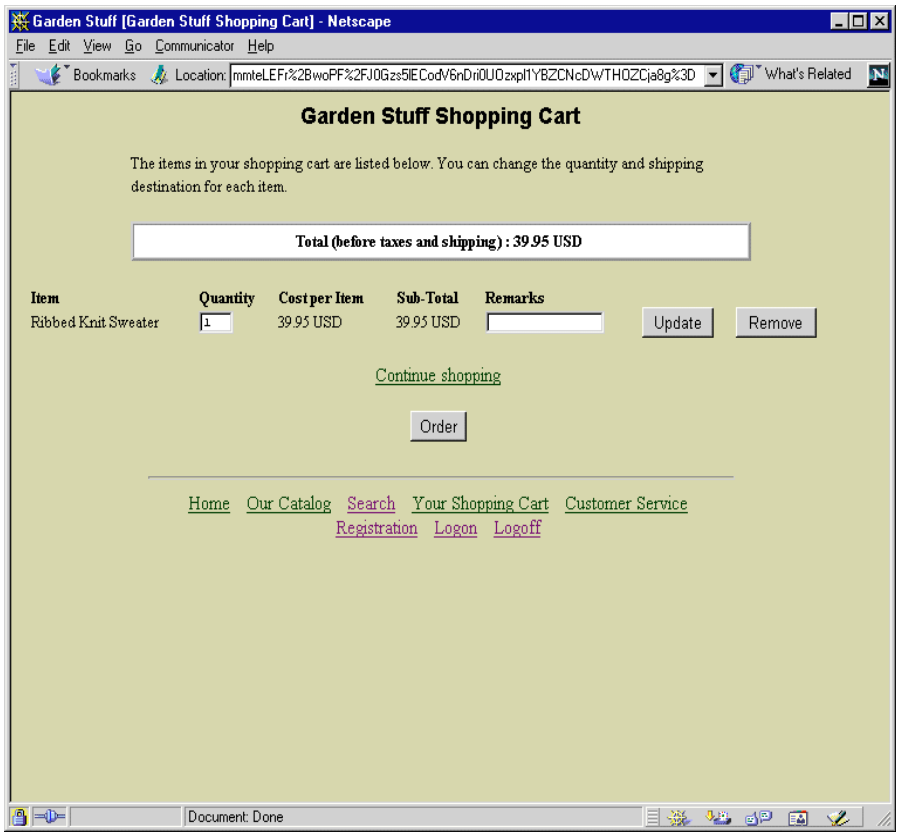
SKU1432
Size:XS, S, M, L, XL
Color:Off-White, Hunter Green, Dark Blue

Soft 100% cotton for an all-over comfortable feel. Machine wash. On the Product page, please specify size and color.

Product Price : 39.95 USD

Figure 11. Product page of the CSP store model

In the product page, the shopper adds the item that he/she want to purchase in the shopping cart by clicking the **Order** button. The shopping cart page allows quantities to be updated or item to be deleted. The shopping cart page follows:


Garden Stuff [Garden Stuff Shopping Cart] - Netscape

Garden Stuff Shopping Cart

The items in your shopping cart are listed below. You can change the quantity and shipping destination for each item.

Total (before taxes and shipping) : 39.95 USD

Item	Quantity	Cost per Item	Sub-Total	Remarks
Ribbed Knit Sweater	1	39.95 USD	39.95 USD	

Update
Remove
Continue shopping
Order
Home Our Catalog Search Your Shopping Cart Customer Service
Registration Logon Logoff
Document: Done

Figure 12. Shopping cart page of the CSP store model

When the shopper decides which items will be purchased and clicks the **Order** button in the shopping cart page, they move into the checkout flow. The checkout flow consists of three or four pages as shown in Figure 9 on page 26.

On the order billing address page, the shopper enters the address which the bill is delivered to. And then if the shipping address is the same as the billing address, the shopper goes to the order payment information page or if not, goes to the order shipping address page. For the registered shopper, the order billing address page shows the page filled with the registered billing address.

Finally payment information must be entered in the order payment information page. NCHS supports off-line payment in addition to on-line payment using the SSL protected form. So the merchant can provide one or two payment methods according to their payment management policy. In the case of on-line payment, the shopper must select the brand of card with which to pay and enter the card number and the expiry date of the card in the order payment information page. In the case of off-line payment, the merchant displays the contact points such as e-mail address or toll free phone number so that the shopper can contact the merchant for providing payment information.

After all the processes are completed, the order confirmation page shows the confirmation message for the order including billing address, ordered items and their quantities, total charge and delivery information.

3.3.2 Customizing the default shopping flow

We explained the default shopping and checkout flow in the previous section. Net.Data macro files for the default shopping and checkout flow are in the following directories:

- macro/en_US/CSPstoremodel
- macro/en_US/category/CSPstoremodel
- macro/en_US/product/CSPstoremodel

Net.Data macro files for each store are in the following directories:

- macro/en_US/*id*
- macro/en_US/category/*id*
- macro/en_US/product/*id*

The Net.Data macro files for each store just includes the CSPstoremodel file like below:

```
.../NetCommerce3/macro/en_US/category/1478/csp_cat.d2w

%include "1478/gardenstuff.inc"
%include "category/CSPstoremodel/csp_cat.d2w"
#
```

A CSP can change the default shopping and checkout flow for the site by modifying Net.Data macro files under the CSPstoremodel directories. And the CSP can also change the flow of specific store by modifying Net.Data macro files in the specific store directories. In that case the modification is applied only that specific store.

In this section, we will show how to change the default store flow by modifying Net.Data macro file. We will replace the **Order** button in the category items page with the **Quick Order** button and let the shopper go from category items page to shopping cart page directly.

The only thing that the shopper might do for ordering in the product page is inserting the attributes of the item which he/she will purchase. And the product page provides no more information about the item than the category items page does. So we can replace the **Order** button without losing any information with the **Quick Order** button. After customization, the shopper can see the product page by clicking the image of the item. You can refer to screen shots of the category items page and the product page in Figure 10 on page 27 and Figure 11 on page 28.

The new shopping flow is as like this:

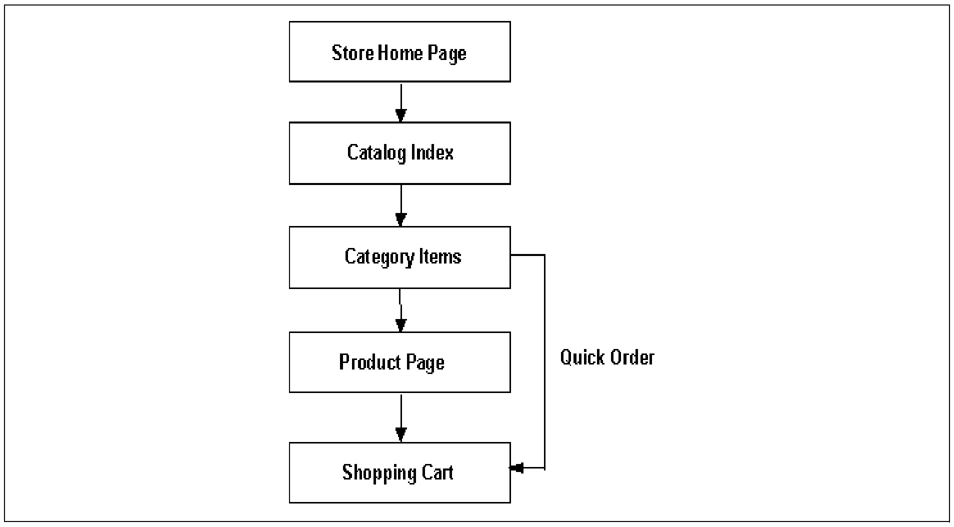
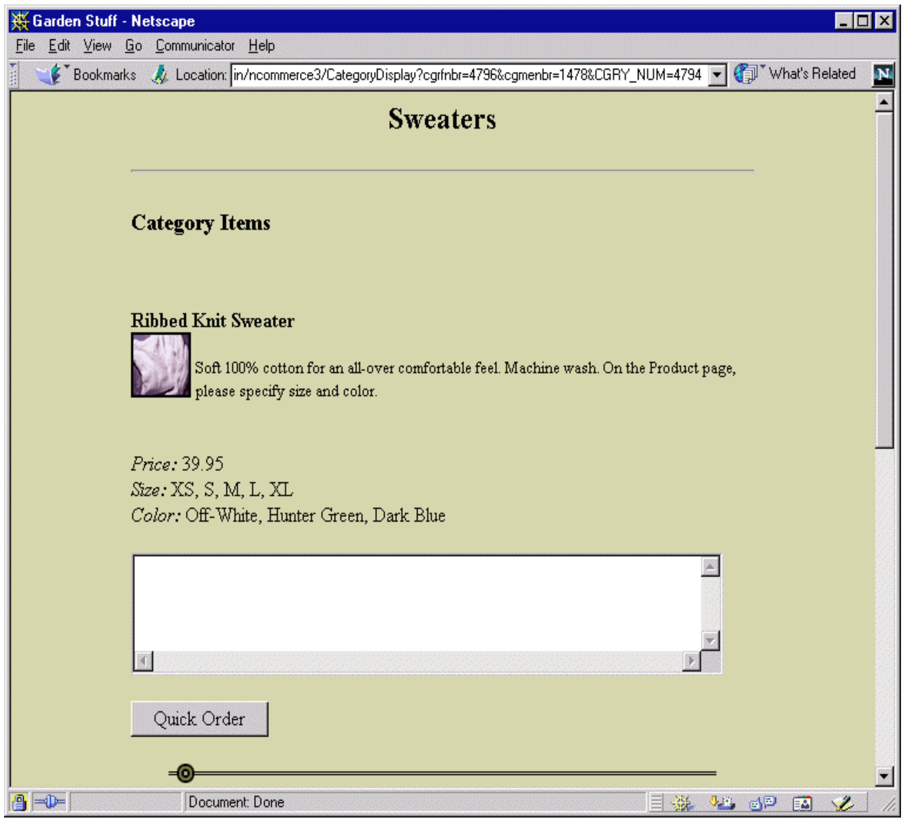


Figure 13. Shopping flow after adding Quick Order button

The figure of the category items page after customizing will be as like below:


Garden Stuff - Netscape
File Edit View Go Communicator Help
Bookmarks Location: in/ncommerce3/CategoryDisplay?cgrfnbr=4796&cgmenbr=1478&CGRY_NUM=4794
What's Related

Sweaters

Category Items

Ribbed Knit Sweater

Soft 100% cotton for an all-over comfortable feel. Machine wash. On the Product page, please specify size and color.

Price: 39.95
Size: XS, S, M, L, XL
Color: Off-White, Hunter Green, Dark Blue

Quick Order
Document: Done

Figure 14. Category items page after replacing Order button with Quick Order button

If the shopper want to purchase the item, he/she puts the item in the shopping cart by entering attributes of the item in the comment area and clicking the **Quick Order** button. By replacing the **Order** button with the **Quick Order** button, the CSP can provide more convenient shopping process.

You can customize the category items page of the default store flow by modifying macro/en_US/category/CSPstoremodel/csp_cat.d2w macro file.

The things that we have to do for customizing are as follows:

- Delete the **Order** button that submits a data set to ProductDisplay command. The NCHS command ProductDisplay displays a product page.

- Insert the **Quick Order** button that submits a data set to OrderItemUpdate command. The NCHS command OrderItemUpdate updates or creates a shipping record.

First of all we have to delete the **Order** button from the category items page. Open the text translation file named as macro/en_US/CSPstoremodel/translation_text.inc. You can see that the variable BUT_ORDER represents the text "Order" in the macro file.

%define {

```
        BUT_CATINDEX            = "Catalog Index"
        BUT_CHANGE              = "Change"
        BUT_CONTINUEORDER       = "Continue"
        BUT_FINDLOGONID         = "Find"
        BUT_HOME                = "Home"
        BUT_LOGON               = "Logon"
        BUT_ORDERITEMS          = "Order"
        BUT_ORDER               = "Order"
        BUT_REG_UPDATE          = "Update Registration"
```

Open the file macro/en_US/category/CSPstoremodel/csp_cat.d2w using a text editor and find the command including the variable BUT_ORDER. You can see three places in the function DISPLAY_PRODUCT_LIST. The **Order** button was used to submit a form data set to ProductDisplay command. The command ProductDisplay displays a product page, so we don't need to execute that command in the customized category items page.

We can comment out the form text including the ProductDisplay command and the **Order** submit button. Following is the example, and you can apply the same method in the other two form text in the function DISPLAY_PRODUCT_LIST.

```
%IF ( pre_rrfnbr != "null" && V_prrfnbr != prr_no )
<!-- <FORM ACTION="/cgi-bin/ncommerce3/ProductDisplay" METHOD="post">
<INPUT TYPE=hidden NAME="prmenbr" VALUE="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="prrfnbr" VALUE="$(pre_rrfnbr)">
<INPUT TYPE=hidden NAME=product_rn VALUE=$(pre_rrfnbr)>
<INPUT TYPE=SUBMIT VALUE="$(BUT_ORDER)">
</FORM> -->
```

Next we have to insert the form text executing the command OrderItemUpdate in the three place that we commented out the form text executing the command ProductDisplay. In the form text the **Quick Order** button submits a form data set to OrderItemUpdate command. The example of the form text to be inserted is as like below:

```
<FORM NAME="process" ACTION="/cgi-bin/ncommerce3/OrderItemUpdate"
Method=get>
<INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>
<INPUT TYPE=hidden   NAME=product_rn   VALUE=$(pre_rrfnbr)>
<INPUT TYPE=hidden   NAME=quantity     VALUE=1>
<INPUT TYPE=hidden   NAME=url
VALUE=/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchantRefNum)>
<TEXTAREA NAME="comment" ROWS="4" COLS="60"></TEXTAREA>
<br><br>
<INPUT TYPE=SUBMIT VALUE="Quick Order">
</FORM>
```

The command OrderItemUpdate updates or creates a shipping record. The data set for the command OrderItemUpdate includes merchant reference number, product reference number, quantity of the item (in this case, using default quantity), url that is called when the command successfully completes (in this case, OrderItmeList command) and a comment to be included with the order.

You can see the overall code for customized macro/en_US/category/CSPstoremodel/csp_cat.d2w file in Appendix C, “Net.Data macro for the category items page” on page 65.

3.4 Adding a new function to NCHS-"Gift message" exmaple

In this section we will go into detail about how a new function could be added to NCHS. We will do this by creating an example and then make the necessary changes to the merchant tool, the checkout flow and the database.

Some shops sell goods that are appropriate as gifts. Such shops may want to offer an additional service by allowing shoppers to enter personalized gift messages when they order. However, not all shops have a business model where gift messages are appropriate. Hence, allowing shoppers to enter gift messages should therefore be offered as an option to merchants.

Adding a feature like a gift message to NCHS, requires a number of modification. This becomes clear when we view the list of things that must be implemented in order for this feature to become active:

- The merchant must be able to enable and disable the gift message feature.

- Instructions on how to enter a gift message must be presented to the shopper. The instructions may differ by each merchant and each merchant should therefore be able to enter his own instructions.
- If a merchant has enabled the gift message feature, then the shopper should be presented with this option during the checkout process.
- The gift message must be stored with the order and the merchant should be able to see the message in the merchant tool.

An implicit change not mentioned above is the need for a place in the database to store information about which of the merchants has enabled the gift message feature and what are their related instructions. A place to store the actual gift messages is also required.

The following sections will explain how the gift message feature is implemented in the various parts of NCHS. The first item we modify is the merchant tool, where we will let the merchant enable/disable our new feature.

3.4.1 MultiPurpose Code Generation language (MPG)

Disclaimer

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 Net.Commerce Hosting Server. IBM does not guarantee any migration path for changes made to files in relation to MPG..

Many of the windows in the merchant tool are generated by the MultiPurpose Generator (MPG) within NCHS. The MPG is a utility used to generate output based on two components: the model (a Java class) and a template (a text file). In NCHS the template describes the output generated for the browser which is HTML. All the MPG templates are text files and can be modified with a standard text editor. The Java classes that makes up the model part of the page generation are compiled codes and can not be modified. This fact creates some limitations to what can be customized in the merchant tool.

The MPG models for all the MPG pages exist within the NCHS Java framework and since it is not disclosed how these models are implemented, we can not create new ones. However, we can still modify the existing pages, as MPG allows some simple programming in the template part of a page. The graphical design can also be changed. We will take advantage of these options as we implement the gift message feature.

The template files are interpreted by the MultiPurpose Generator whenever they are read and parsing the templates takes a while. To avoid this overhead, the parsed version of the templates are cached and then reused for subsequent reads. This scheme only works if the templates are not modified. Hence, modifying the templates on a regular basis should be avoided.

Appendix A, “A MultiPurpose Code Generation language” on page 331 is a document that gives an introduction to how MPG works. This document is not exhaustive and is only included to help the reader getting a better understand of MPG. We are not doing much MPG programming in this book and just a basic understanding of MPG is required to read and understand the examples we provide.

Please note that when we refer to MPG files in the rest of this book, we mean the template files.

3.4.2 Add/Remove a menu item in the merchant tool

To allow merchants to see the new gift message feature and then use it, we have to modify the merchant tool by adding a new menu item. We will call this menu item **gift message** and we would like to insert it between **open/close store** and **upload files** on the Set Up Your Store page (see Figure 8 on page 22 to refresh you memory of the menu layout).

The menu to the left is generated from an XML file called merchantTool.xml located in /Tools/xml/nchs/mtool/. This file covers all the left hand menus on all pages in the merchant tool. Fortunately it is not that difficult to find the right spot for out new menu item.

We will not print the entire merchantTool.xml file due to its size. Instead use a text editor to open the file and search for OpenCloseStore. You should then be located in the file as shown below:

```
<link name  = "openCloseStore"
url   = "/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.StoreState"
users = "storeAdmin" />
<link name  = "uploadFiles"
url = "/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.SplashScreen&amp;
window = FileUpload"
users = "storeAdmin,catalogAdmin" />
```

If you scroll up and down in the file you will notice that every menu item has its own entry as a link tag.

All we have to do is to create a link tag with parameters for the new gift message menu item and place it in merchantTool.xml file between openCloseStore and uploadFiles. The code we will insert looks like this:

```
<link name  = "giftMessage"
url   = "/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/gftmsg.d2w/
report?merfnbr=" + $env.merchant_id$"
users = "storeAdmin" />
```

Insert this code between the openCloseStore and advancedFeatures links and it will look like this:

```
<link name  = "openCloseStore"
url   = "/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.StoreState"
users = "storeAdmin" />
<link name  = "giftMessage"
url   = "cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/gftmsg.d2w/
report?merfnbr=" + $env.merchant_id$"
<link name  = "uploadFiles"
url   = "/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.SplashScreen&amp;
window=FileUpload"
users = "storeAdmin,catalogAdmin" />
```

The first line defines variable that contains the name used on the link. The second line defines the action of the link. And the third line defines the users allowed to see this link.

Save the merchantTool.xml file.

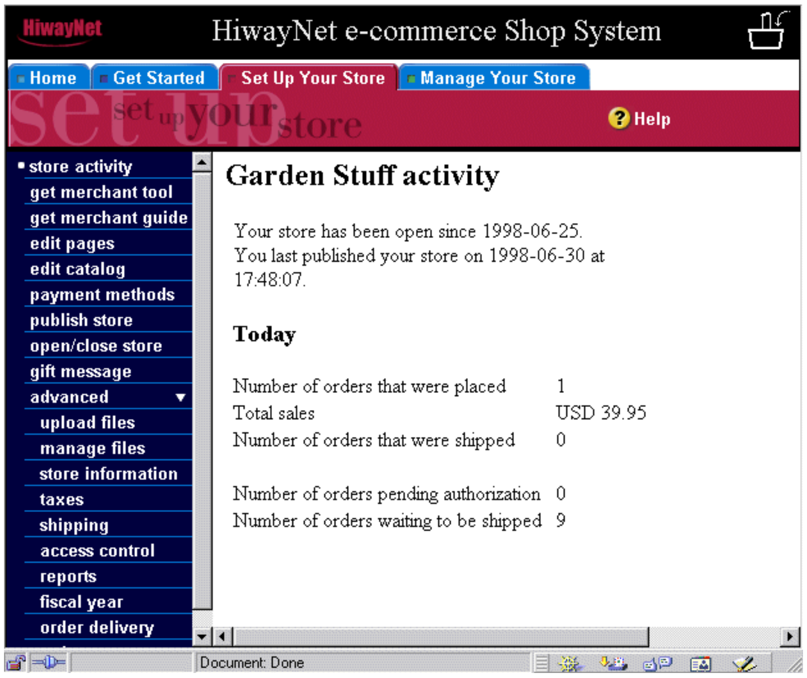
Since the link name reference in the merchantTool.xml file refers to a variable that holds the name of the link, we must provide the name of the link to that variable. The file /NetCommerce3/Tools/lib/nchs.jar contains many files and among them is mtoolNLS.properties. Extract mtoolsNLS.properties from nchs.jar and open the file for editing. Enter the following line in the '# Merchant tool set up your store links' section:

```
# Merchant tool set up your store links
storeActivity=store activity
getMerchantTool=get merchant tool
getMerchantGuide=get merchant guide
publishStore=publish store
paymentMethods=payment methods
editPages=edit pages
editCatalog=edit catalog
openCloseStore=open/close store
giftMessage=gift message
```

Save the file and replace into the nchs.jar file, making sure the path is saved as well.(You may have to stop the Domino Go Webserver to replace the file, then restart it).

View the new menu in your browser. It should look something like Figure 15. The new menu item is right between **open/close store** and **upload files**.

Notice on Figure 15 that we have gotten a vertical scroll bar next to the left hand menu. This is because we expanded the menu outside the window and thus got a scroll bar. To get rid of the scroll bar we could either ask the user to expand the window or change the windows default opening size.



Garden Stuff activity

Your store has been open since 1998-06-25.
You last published your store on 1998-06-30 at 17:48:07.

Today

Number of orders that were placed	1
Total sales	USD 39.95
Number of orders that were shipped	0
Number of orders pending authorization	0
Number of orders waiting to be shipped	9

Figure 15. Merchant tool with new menu item

3.4.3 Change the size of the merchant tool window

The merchant tool is by default opened when a user tries to create a store or manage a store. The two files that control the size of the merchant tool window are create_store.html and manage_store.html. These files are located in the directory html/en_US/cspsite.

```
# ls /usr/lpp/NetCommerce3/html/en_US/cspsite/ *_store.html
buy_store.html     create_store.html  manage_store.html  view_store.html
#
```

Edit the create_store.html file and look for a line that begins with

```
var target = window.open("", "MerchantTool", "resizable=yes, ...
```

You could also just search for *height* as that would bring you to the same line and place your cursor right where you want it.

The entire line looks like this:

```
var target = window.open("", "MerchantTool", "resizable=yes,scrollbars=
yes,status=yes,width=750,height=500,screenX=0,screenY=0,left=0,top=0");
```

We want to change the height of the merchant tool window from 500 pixels to 610 pixels, so the height=500 parameter should be changed to height=610.

```
var target = window.open("", "MerchantTool", "resizable=yes,scrollbars=
yes,status=yes,width=750,height=610,screenX=0,screenY=0,left=0,top=0");
```

Save the create_store.html file with the new changes. The next step is to modify the manage_store.html file. Its the same process.

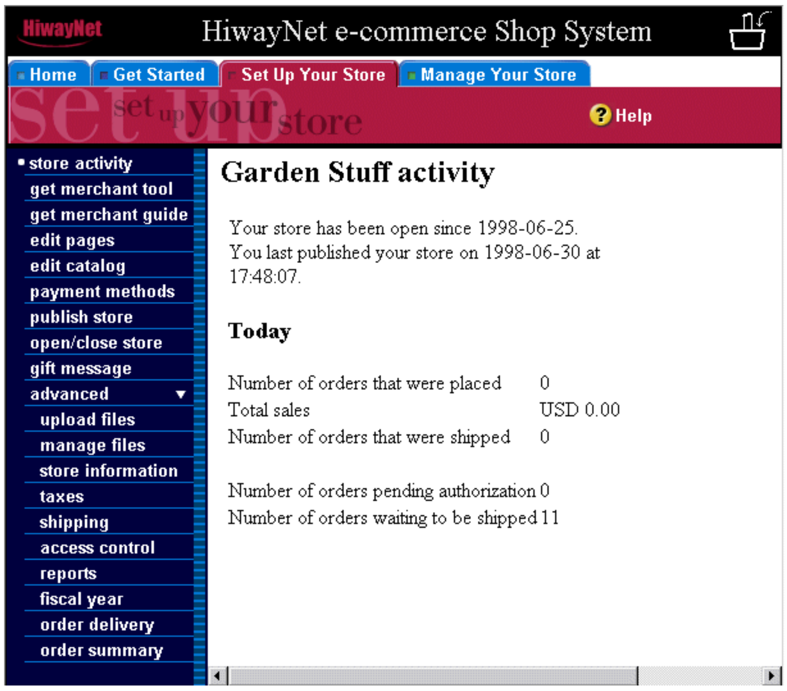
In manage_store.html search for *height* or look for this line:

```
window.open("http://" + window.location.hostname +
"/servlet/MerchantAdmin?GOTO=Banner&body=LogonPage", "MerchantTool",
"resizable=yes,scrollbars=yes,status=yes,width=750,height=500,screenX=0,
screenY=0,left=0,top=0");
```

Change the height=500 parameter to height=610 as

```
window.open("http://" + window.location.hostname +
"/servlet/MerchantAdmin?GOTO=Banner&body=LogonPage", "MerchantTool",
"resizable=yes,scrollbars=yes,status=yes,width=750,height=610,screenX=0,
screenY=0,left=0,top=0");
```

Save the manage_store.html. Select manage store from the from the cspsite Web page, log in, select the **Set Up Your Store** page. The vertical scroll bar should have disappeared as shown on Figure 16 on page 40.



Garden Stuff activity

Your store has been open since 1998-06-25.
You last published your store on 1998-06-30 at 17:48:07.

Today

Number of orders that were placed	0
Total sales	USD 0.00
Number of orders that were shipped	0

Number of orders pending authorization 0
Number of orders waiting to be shipped 11

Figure 16. The merchant tool with no vertical scroll bar

Notice that there still is a horizontal scroll bar, but this is only because we have shrunk the image horizontally to make it look better in this book.

3.4.4 A new Net.Data macro for the merchant tool

As we add a new menu item to the menu of the merchant tool we also have to add a page for the right hand window of the merchant tool. The right hand window is where NCHS presents and receives information to and from the merchant.

In section 3.4.2, “Add/Remove a menu item in the merchant tool” on page 36 we created a new menu item called "gift message" and it will launch a Net.Data macro called gftmsg.d2w when it is selected.

The information we want to collect with this macro is whether or not the merchant wants to enable the gift message feature and the gift message instructions the merchant presents to the shopper. To get this information we need at checkbox field for the enable/disable function and a text field for the instructions. Our form should also have a submit button so the merchant can submit the information to our system.

We need something to receive the data from the submit button. The usual way with Net.Commerce is to have a command receive and process the data before launching a new Net.Data macro back to the browser. Since our data is very simple and we only use our own table, we will not create a new command but just use a Net.Data macro to receive and store the data. Please note that the recommended procedure is to use a Net.Commerce command to process the data.

Before we create the Net.Data macro we need a place to store the data we receive. When a merchant enables the gift message feature then the checkout process for that merchant's store should automatically change to support getting the gift message from the shopper. To do that we need an enable/disable flag and a place to store the gift message instructions. The database is an obvious choice.

There are a couple of ways to store new data in the Net.Commerce database:

1. Add new fields to the MERCHANT or MCSPINFO table as there are already one row for each merchant (and our data must be related to a specific merchant). This approach has a huge drawback; if or when we decide to upgrade our installation to the next version of NCHS, we are very likely to encounter a number of difficulties upgrading the database scheme because we changed it. We can not expect the next version of NCHS to handle the changes we make to the NCHS tables.
2. Use some of the "reserved for merchant customization" fields that are available in the NCHS tables. There are actually two fields in the MERCHANT table we could for our purpose. We will use this approach in a later step, but for now we will use the third option.
3. Create a new table with the fields we need. This will give us the most freedom but also some challenges since we have to maintain the relationships to the rest of the database and we also have to handle cleaning of unused data. The cleaning can be done automatically by expanding the referential integrity relationships already implemented in the database.

Option 1 should be avoided in any case, so we can choose between option 2 and 3 . We will choose option 3 to show how to implement your own table, but in section 3.4.5, “Modify the checkout flow” on page 44, we will implement option 2 and use the "reserved for merchant customization" fields.

Apart from an enable/disable flag and some space for instructions we also need to relate our data to a certain merchant. The below table shows the three fields contained in our new table.

Table 1. MADDFEATURE: Merchant Additional Feature Table

Name	Type	Description
MAMENBR	INTEGER NOT NULL	Merchant reference number. This is a foreign key that reference column MERFNBR in table MERCHANT.
MAGFTMSG	SMALLINT	0 - Gift message feature disabled 1 - Gift message feature enabled
MAGFTTXT	VARCHAR(256)	The merchants gift message instructions to shoppers.

The following is a description of how to create the MADDFEATURE table in the database.

1. Log in as the database instance owner (db2inst1 in our example).
2. We will create a small script file to create the table. Actually the script only contains one command, but its easier to edit a script than to type the entire command again if we want to make changes.

   Create a file called giftmessage.db2.sql with a text editor.
3. Add the following text to the script file:

```
create table maddfeature
             (
             mamenbr      integer not null,
             magftmsg     smallint,
             magfttxt     varchar(256),
             constraint   p_maddfeature primary key (mamenbr),
             constraint   fme_maddfeature foreign key (mamenbr)
                          references merchant (merfnbr)
                          on delete cascade
             );
```

4. Save the script file and exit the text editor.
5. Connect to the database by issuing the following command:

```
db2 connect to mser
```

   Our database is called mser which is the Net.Commerce default name.
6. Run the script to create the table:

```
db2 -tvf giftmessage.db2.sql
```

When you executes the script it should look something like this:

```
db2 -tvf gift_message.db2.sql
create table maddfeature ( mamenbr integer not null, magftmsg smallint,
magfttxt varchar(256), constraint p_maddfeature primary key (mamenbr),
constraint fme_maddfeature foreign key (mamenbr) references merchant
(merfnbr) on delete cascade )
DB20000I  The SQL command completed successfully.
```

7. If you for some reason would like to delete the table (and all its contents) then use this command:

   ```
   db2 drop table maddfeature
   ```

The database table is now created and we can begin to develop the two Net.Data macros. Figure 17 indicates how the flow of the Net.Data macros should be.

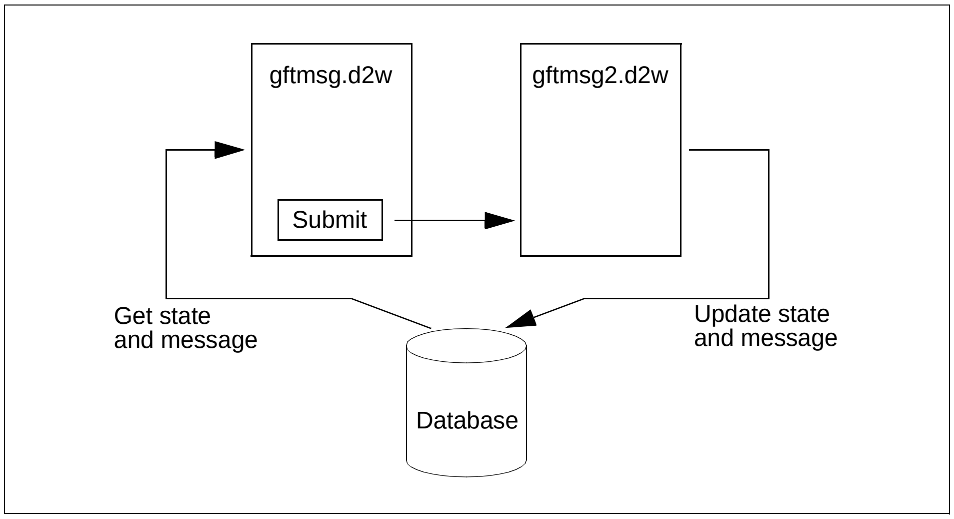


Figure 17. Merchant tool gift message flow

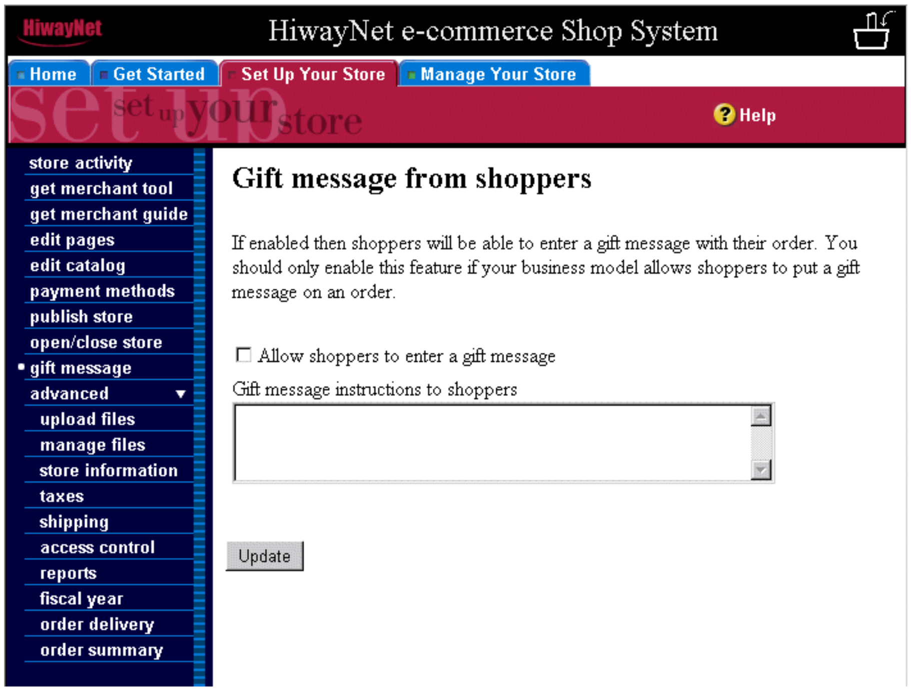
The gftmsg.d2w macro reads the current status (whether the gift message feature is enabled or disabled) and the instruction from the database. This is presented in the form to the merchant. The merchant can alter the settings and submit the changes to gftmsg2.d2w. The gftmsg2.d2w macro receives the data and updates the database. As a confirmation the gftmsg2.d2w macro presents the updated data to the merchant.

To be consistent with the merchant tool we place our new Net.Data macros in the directory macro/en_US/ncadmin/storemgr. We used the macro cspGenRptStore.d2w as a starting point for developing our own macros, instead of beginning from scratch. The code for the two macros can be found in Appendix A., “Net.Data macros for the merchant tool” on page 45.

The full URL to launch the gftmsg.d2w macro is shown below. Log in as a merchant first or you will get an access denied message.

```
http://www.hostname.com/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/gftm
sg.d2w/ report?merfnbr=" + env.merchant_id
```

The URL above is used in the merchant tool as described in section 3.4.2, “Add/Remove a menu item in the merchant tool” on page 36. Figure 18 shows what the gift message window looks like in the merchant tool.



Gift message from shoppers

If enabled then shoppers will be able to enter a gift message with their order. You should only enable this feature if your business model allows shoppers to put a gift message on an order.

☐ Allow shoppers to enter a gift message

Gift message instructions to shoppers

Figure 18. The new gift message window

3.4.5 Modify the checkout flow

In the previous section the merchant tool and database was modified to handle the gift message feature. If the merchant enables the gift message feature then shopper should get a field where they can enter a gift message.

The gift message instructions from the merchant should also be displayed to the shopper.

The most natural place to display and collect the gift message would be during the checkout process. We decided to expand the "order payment information" page with a gift message section. The macro used to display this page is called ord_pay.d2w.

```
.../NetCommerce3/macro/en_US/CSPstoremodel/addradd.d2w:
logon_pwdchange.d2w   ord_status.d2w
address.inc           main.d2w              ord_update.d2w
br_turnoff.d2w        navbar.inc            pwdresform.d2w
cservice.d2w          ord_billto.d2w        reg_new.d2w
err_addr.d2w          ord_details.d2w       reg_update.d2w
err_dopay.d2w         ord_ok.d2w            search_display.d2w
err_reg.d2w           ord_pay.d2w           translation_text.inc
logon.d2w             ord_pay.d2w.org
logon_idforgot.d2w    ord_shipto.d2w
```

We modified the ord_pay.d2w macro by adding a function called GET_GIFTMESSAGE(). This function basically checks if the merchant has enabled the gift message feature, and if the merchant has, then it adds some HTML to the generated page. The source code for our new version of ord_pay.d2w can be found in Appendix B., “Net.Data macro for the checkout flow” on page 51.

We query the database to find out whether or not the merchant has enabled the gift message feature and to get the instructions. The query is done against the table we created in section 3.4.4. After the query we have the necessary information to determine whether or not we should add the extra HTML for the gift message feature.

The new HTML contains the gift message instructions and a new field called *giftmessage*. The giftmessage field is of the HTML type called TEXTAREA and is not known to NCHS. What this means is that the command that handles the output of the ord_pay.d2w macro is called `OrderProcess` and it does not know about our giftmessage field. Hence, it will not do anything with it. What we need is something that handles our giftmessage field and stores it with the current order. To do that we must enhance the `OrderProcess` command.

There is a description of how all the NCHS commands work in the manual called "Commands, Tasks, Overridable Function and Database Tables". The `OrderProcess` command calls three process tasks, the last of which is

EXT_ORD_PROC. The manual states that the EXT_ORD_PROC task is used to "perform additional processing". The overridable function (OF) DoNothingNoArgs is by default assigned to this task. The DoNothingNoArgs OF does nothing, as the name suggests. We can write a new OF which is then executed by the EXT_ORD_PROC task.

The new OF, which is called AddGiftMessage, takes the gift message from our HTML form field giftmessage and stores it with the order. A suitable place in the database is needed to store the message. Again we look in the manual "Commands, Tasks, Overridable Function and Database Tables" to find a space for our gift message. Since we would like to store the gift message with the order it belongs to we look in the ORDERS table. It has a field called ORFIELD3 which is "Reserved for merchant customization" and is of type VARCHAR(254). This field suits our purpose and we will use it to store the gift message for an order.

Its beyond the scope of this book to explain how to write an OF, instead we refer to the manual "Commands, Tasks, Overridable Functions and the E-Commerce Framework". It describes the Net.Commerce v3.x internal programming model and is a *must have* for everybody interested in customizing Net.Commerce. It is updated every 2-3 months and should therefore be downloaded on a regular basis. When this book was written it could be downloaded from

```
http://www.software.ibm.com/commerce/net.commerce/downloads/aix/v3.1.2/dow
ndocs.html
```

However, as everything on the Web changes often, you may just want to go to

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

and then go into the download section and find the documentation section for Net.Commerce.

The C++ source code for the OF we wrote can be found in Appendix D., “Source code for AddGiftMessage OF” on page 73. We made and compiled the code on AIX and NT, but it should not be a problem to move the OF to other platforms. The makefile is also in the same appendix.

When you have compiled the code you should end up with a file called libaddgiftmessage.a for AIX or addGiftMessage.dll for NT. It must be copied to the bin directory of Net.Commerce.

For AIX, the access permissions should be set accordingly. Stop NCHS before continuing with the following commands:

```
# cp -f libaddgiftmessage.a /usr/lpp/NetCommerce3/bin/libaddgiftmessage.a
# chmod 444 /usr/lpp/NetCommerce3/bin/libaddgiftmessage.a
# chown bin.bin /usr/lpp/NetCommerce3/bin/libaddgiftmessage.a
```

The -f in the cp command activates the force feature. If you have already copied the libaddgiftmessage.a to the bin directory and Net.Commerce use it, it is sometimes locked even when Net.Commerce is stopped. The force feature overwrites the file even if its locked, so be careful.

After coping the file you need to make Net.Commerce aware of it, and make it the new OF for the EXT_ORD_PROC task. This kind of information is kept in the database and we refer to the manuals *"Commands, Tasks, Overridable Functions and the E-Commerce Framework"* and *"Commands, Tasks, Overridable Function and Database Tables"* for a complete description. The SQL we execute to activate our OF is put into a file called AddGiftMessage.db2.sql. The contents is shown here:

```
delete from ofs where name='AddGiftMessage';

insert into ofs (refnum , dll_name, vendor, product, name, version,
description) values
((select max(refnum) from ofs) + 1, 'libaddgiftmessage.a', 'IBM ITSO',
'NC', 'AddGiftMessage', 1.0 , 'Shoppers can provide a gift message with
their order');

delete from task_mer_of where task_rn=(select tkrfnbr from tasks where
tkname='EXT_ORD_PROC') and merchant_rn is null;

insert into task_mer_of (task_rn, of_rn) values (
(select tkrfnbr from tasks where tkname='EXT_ORD_PROC'),
(select refnum from ofs where name='AddGiftMessage')
);

commit;
```

The file AddGiftMessage.db2.sql is executed by the command

```
db2 -tvf AddGiftMessage.db2.sql
```

You should of course be logged in as the database instance owner (db2inst1) and have connected to the database, when you execute the above command.

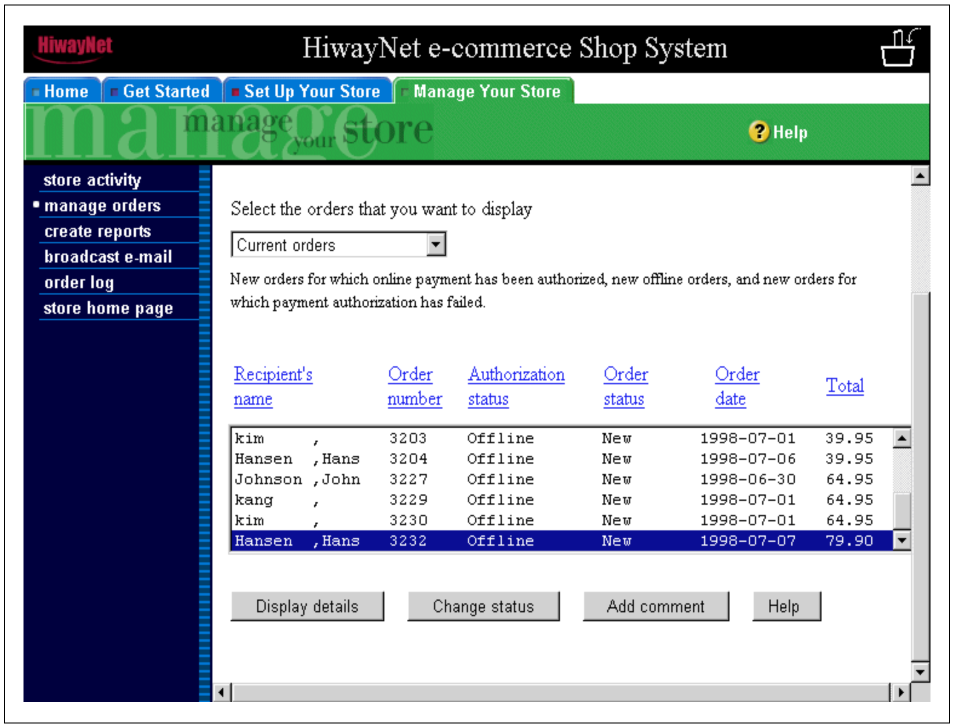
3.4.6 Adjust the order details page

In the previous sections, we added the gift message menu in the merchant tool and changed the checkout flow. So the merchant becomes able to enable and disable the gift message feature and the shopper becomes able to insert gift message in his/her order information.

In this section, we will modify the **manage orders** menu in the merchant tool so that the merchant can manage the gift message that the shopper inserted in his/her order information.

Using the **manage orders** feature in the **Manage Your Store** tab, the merchant can do the following tasks as you can see in Figure 19:

- Change the status of an order
- Display the details of an order
- Add a comment to an order
- Remove a comment to an order



Select the orders that you want to display

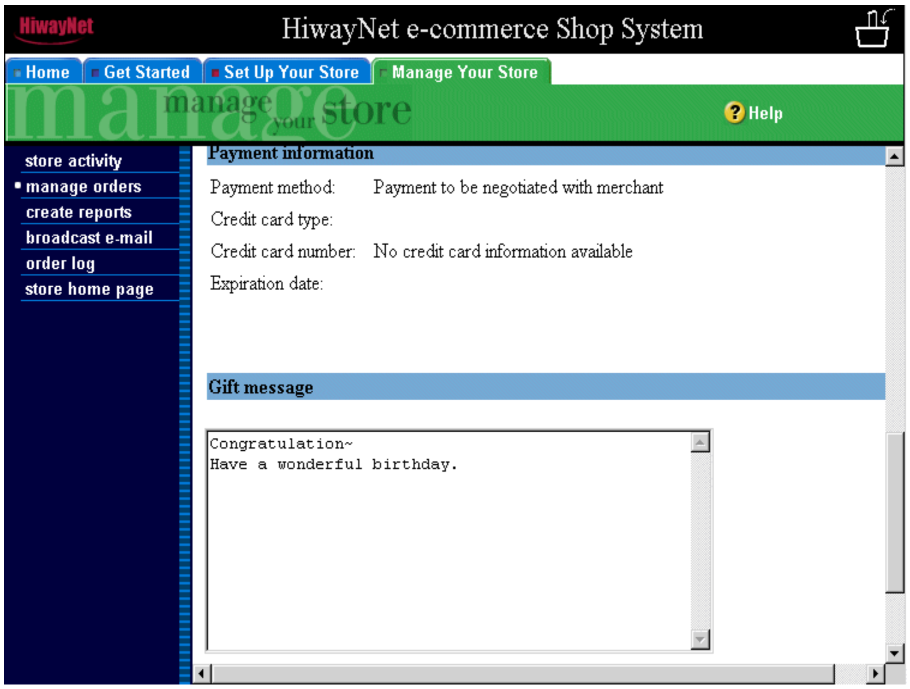
New orders for which online payment has been authorized, new offline orders, and new orders for which payment authorization has failed.

Recipient's name	Order number	Authorization status	Order status	Order date	Total
kim ,	3203	Offline	New	1998-07-01	39.95
Hansen ,Hans	3204	Offline	New	1998-07-06	39.95
Johnson ,John	3227	Offline	New	1998-06-30	64.95
kang ,	3229	Offline	New	1998-07-01	64.95
kim ,	3230	Offline	New	1998-07-01	64.95
Hansen ,Hans	3232	Offline	New	1998-07-07	79.90

Figure 19. the ***manage orders*** *feature in the* ***Manage Your Store tab***

If you select an order in the order list and then click the **Display details** button, you can see the order details page of the selected order. The order details page displays the order, address and payment information. We will append the gift message field at the end of order details page so that the merchant can see and manage the gift message of the selected order.

The order details page looks like Figure 20 after appending the gift message. The order details page will display the gift message field whenever the shopper inserted the gift message in his/her checkout flow of the selected order. If there is no gift message for the selected order, the order details page will not display the gift message field.



Payment information

Payment method: Payment to be negotiated with merchant
Credit card type:
Credit card number: No credit card information available
Expiration date:

Gift message

```
Congratulation~
Have a wonderful birthday.
```

Figure 20. The order details page displaying the gift message

To append the gift message field in the order details page, we must know the template file for the order details page. If you explore the .../NetCommerce3/Tools/xml/nchs/mtool/merchantTool.xml file introduced in 3.4.2, “Add/Remove a menu item in the merchant tool” on page 36 using a text editor, you will find that the CHS_OrderProcess command is called when the **manage orders** menu is selected. Open the .../NetCommerce3/Tools/config/nchs/orderMgmtTasks.xml file and search for the task CTnchs.order_mgmt.Main. Then you can see that

.../NetCommerce3/Tools/mpg_templates/nchs/order_mgmt/Main.tem is the template file for the order management page.

Under the function launchOrderDetails you can see it calls for orderDetails which uses the orderDetails.tem template.

We have to do two things in order to append the gift message to the order details page:

- Retrieve the gift message of the selected order from the ORDERS table
- If there is a gift message for the selected order, display the gift message at the end of the order details page

We have to create the procedure in orderDetails.tem that displays the gift message if one exists. The code for the procedure is below:

```
  addGiftMessage()
  {
  Query stmt10
  Var   giftmsg

  giftmsg = ""

  stmt10 = "select ORFIELD3 from ORDERS where ORRFNBR="+
  parameters.selectedOrder

  stmt10 | reset()
  giftmsg += stmt10.ORFIELD3

  if ( giftmsg != "null" ) {
  /*
  <TABLE BORDER=0  COLS=1 WIDTH="602" BGCOLOR="#6699CC" >
          <TR>
             <TD><B>Gift message</B></TD>
          </TR>
  </TABLE>
  <BR>
  <FORM NAME='giftmessage'>
            <TEXTAREA  rows=10 cols=50 WRAP=on>
$giftmsg$
            </TEXTAREA>
  </FORM>
  */
    }
```

We selected ORFIELD3 field of the selected order record from the ORDERS table and stored it to the variable "giftmsg". We described that we use ORFIELD3 for the gift message in the ORDERS table in the previous section.

And it is necessary to insert the statement calling the procedure "addGiftMessage" after the statement that displays the payment information. You can see the whole orderDetails.tem file after modification in the Appendix E., “Template file for the order details page” on page 79.

Now you can test the overall function of the gift message feature. You can enable and disable the gift message feature for the store using the **gift message** menu in the merchant tool. And after that you can create a test order including a gift message, you can see the gift message of that order in the order details page.

3.5 Creating multiple default store layouts

NCHS only provides for one sample store to start with when creating a store. It would be nice to allow merchants to choose among a number of different store layout designs, then let the merchants proceed to create a store based on what was chosen.

As we mentioned in the previous section, when you create a store, NCHS copies the file layout.sdb to site.sdb into the /usr/lpp/NetCommerce/CHS/layout/<env.locale>/ directory. The file layout.sdb is referenced as a hidden variable in the html page produced by the template register.tem. Other .sdb files can be created with the page editor and saved as a variety of layouts. We can then provide a dropdown list box during store creation to let the merchant select the type of store they want to start with. Here is the way to accomplish this:

1. Create a store.
2. Modify the sample pages to create a new template.
3. Save the store but do not publish it. This will create a new site.sdb file in the .../NetCommerce3/CHS/source/*id*/ directory.
4. Repeat steps 1-3 to create as many sample stores as needed. Each new store will create an site.sdb file in the directory named by the merchant reference number.
5. Copy the site.sdb files to the directory .../NetCommerce/CHS/layout/<env.locale>/ and name them an appropriate name for the layout with an .sdb extension.
6. Publish the sample stores.

In this example the new templates we created contained only a different graphic on the first page. This suits our example because it is a simple change that illustrates the different layouts used. However, in a real situation the store layouts could be differentiated by number of pages, types of pages, placement of objects on the pages, and so on, to accommodate the requirements of merchants that may purchase stores in the mall.

The next step is to allow the merchant to choose the template during store creation. To do this we will add a dropdown list to the registration page. There are two ways we might do this. One is modifying the ../NetCommerce3/Tools/mpg_templates/nchs/mtool/register.tem file and the other is to create a register.html static page so that the register.tem file is not needed. However, if we create a new register.html file to use in place of register.tem and an error occurs in the register task, the merchant will be redirected back to the page that template register.tem creates and not our new html page. There is not a way to redirect this error condition. Therefore creating an HTML page to replace register.tem is not an option. We must modify the register.tem.

The template register.tem is used to display the initial form for creating a store. After the screen is displayed, ../NetCommerce3/Tools/mpg_templates/nchs/mtool/register.tem is used again to process the form. We will modify register.tem to display the choices for the default template to use when creating the store.

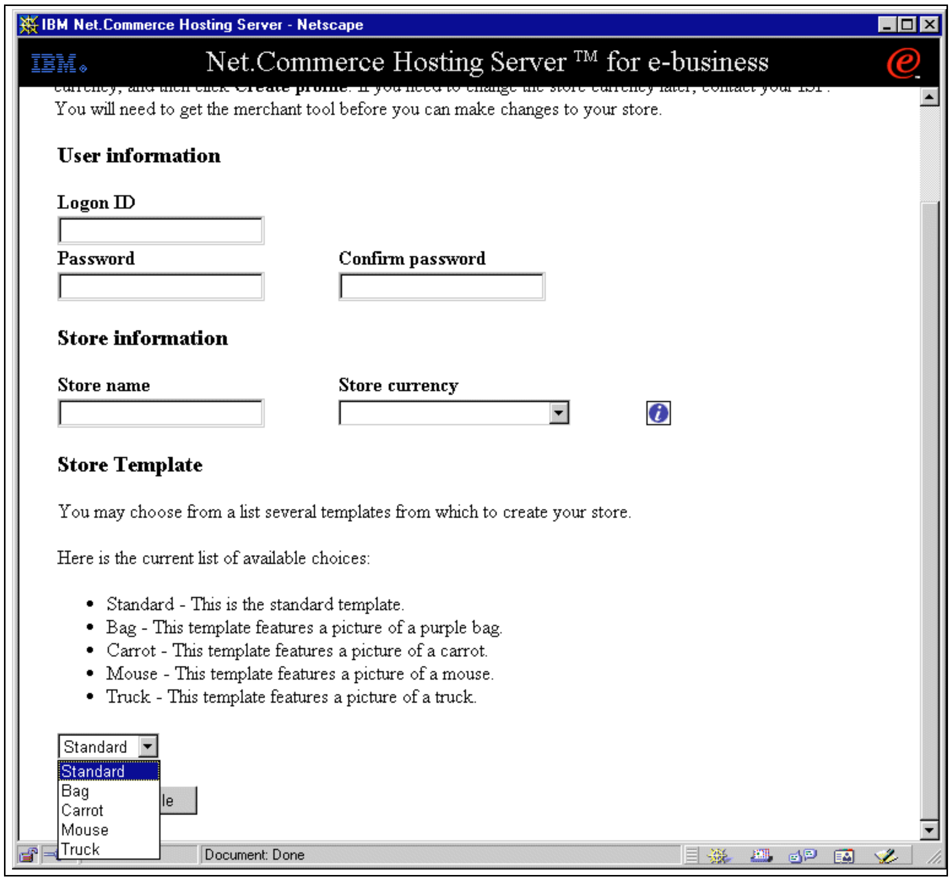
To modify register.tem.

1. Edit .../NetCommerce3/Tools/mpg_templates/nchs/mtool/register.tem. Comment out the line:
   `<INPUT TYPE=hidden NAME="siteDBFile" VALUE="$env.locale$/layout.sdb">`
   then add descriptions of the templates offered and a dropdown box for the variable siteDBfile to list the names of all the template files you created.
2. Save register.tem. You will need to stop then start the web server to reload the new template file.

Here is a sample of code to create the descriptions of the templates and the selection box:

```
<TR><TD WIDTH=30>&nbsp;</TD></TR>
<tr>
<TD WIDTH=30></TD>
<td colspan=2>
<h3> Store Template </h3>
 You may choose from a list several templates from which to create your
store. <p>
```

```
Here is the current list of available choices:
<ul>
<li>Standard - This is the standard template.
<li>Bag - This template features a picture of a purple bag.
<li>Carrot - This template features a picture of a carrot.
<li>Mouse - This template features a picture of a mouse.
<li>Truck - This template features a picture of a truck.
</ul>
<p>
<SELECT NAME="siteDBFile">
               <OPTION VALUE="en_US/layout.sdb">Standard
               <OPTION VALUE="en_US/bag.sdb">Bag
               <OPTION VALUE="en_US/carrot.sdb">Carrot
               <OPTION VALUE="en_US/mouse.sdb">Mouse
               <OPTION VALUE="en_US/truck.sdb">Truck
</select>

</td></tr>
```


IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

currency, and then click **Create profile**. If you need to change the store currency later, contact your ISP.
You will need to get the merchant tool before you can make changes to your store.

User information

Logon ID

Password **Confirm password**

Store information

Store name **Store currency**

Store Template

You may choose from a list several templates from which to create your store.

Here is the current list of available choices:

- Standard - This is the standard template.
- Bag - This template features a picture of a purple bag.
- Carrot - This template features a picture of a carrot.
- Mouse - This template features a picture of a mouse.
- Truck - This template features a picture of a truck.

Figure 21. Create Store Logon Screen.

Now that the merchant can choose which template they want to create their own store from, we need to show the merchant what these choices look like. So the next step is to allow the merchant to view the sample stores.

In the .../NetCommerce3/html/en_US/cspsite/navigation.html file go to the section to view the sample store:

```
<!-- view sample store -->
<TR><TD><IMG NAME="item10" SRC="/CHS/images/bullet_blank.gif" WIDTH=5
HEIGHT=5 ALT="" BORDER=0></TD>
<TD NOWRAP>&nbsp;<FONT FACE="Arial,Helvetica,sans-serif" SIZE=2><B><A
HREF="javascript:go(10,'/cspsite/view_store_banner.html','/cspsite/view_st
ore.html')"
```

```
onMouseOver="on(10); status='view sample store'; return true;"
onMouseOut="off(10); status='';">view sample store</A>
                   </B></FONT></TD></TR>
           <TR><TD></TD>
               <TD><IMG SRC="/CHS/images/separator.gif" WIDTH=122 HEIGHT=1
ALT="" BORDER=0></TD></TR>
```

Change the line:

```
'/cspsite/view_store.html')"
```

to:

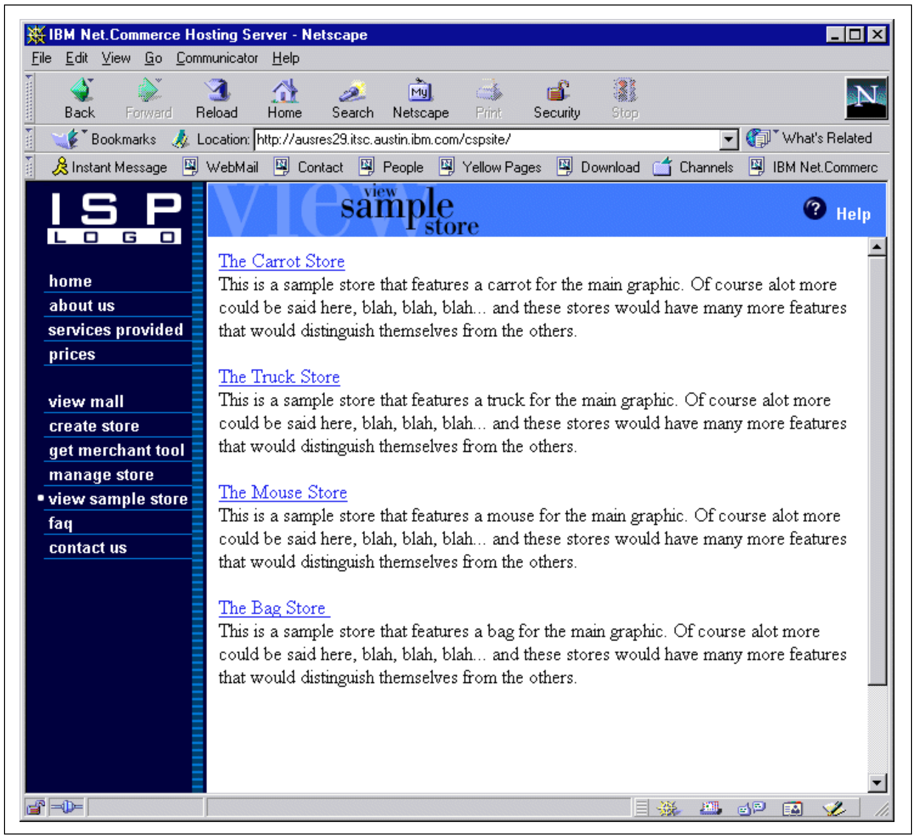
```
'/cspsite/view_samples.html')"
```

and save the file.

This changes the html file used to view the sample stores. Next, create a ../NetCommerce3/html/en_US/cspsite/view_samples.html with the code that describes the sample stores and provides links to those stores. Here is an excerpt from our example:

```
<A href= "/carrot" target="Sample Store"> carrot </A>
<br>This is a sample store that features a carrot for the main graphic.
Of course a lot more could be said here, blah, blah, blah... and these
stores would have many more features that would distinguish themselves from
the others.
<p>
<A href= "/truck" target="Sample Store"> truck </A>
<br>This is a sample store that features a truck for the main graphic.
Of course a lot more could be said here, blah, blah, blah... and these
stores would have many more features that would distinguish themselves from
the others.
<p>
<A href= "/mouse" target="Sample Store"> mouse </A>
<br>This is a sample store that features a mouse for the main graphic.
Of course a lot more could be said here, blah, blah, blah... and these
stores would have many more features that would distinguish themselves from
the others.
<p>
```

The merchant will now be able to view the sample stores and choose one to use when creating their own store.


IBM Net.Commerce Hosting Server - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://ausres29.itsc.austin.ibm.com/cspsite/ What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels IBM Net.Commerc
ISP LOGO
home
about us
services provided
prices
view mall
create store
get merchant tool
manage store
view sample store
faq
contact us
view sample store
Help

The Carrot Store
This is a sample store that features a carrot for the main graphic. Of course alot more could be said here, blah, blah, blah... and these stores would have many more features that would distinguish themselves from the others.

The Truck Store
This is a sample store that features a truck for the main graphic. Of course alot more could be said here, blah, blah, blah... and these stores would have many more features that would distinguish themselves from the others.

The Mouse Store
This is a sample store that features a mouse for the main graphic. Of course alot more could be said here, blah, blah, blah... and these stores would have many more features that would distinguish themselves from the others.

The Bag Store
This is a sample store that features a bag for the main graphic. Of course alot more could be said here, blah, blah, blah... and these stores would have many more features that would distinguish themselves from the others.

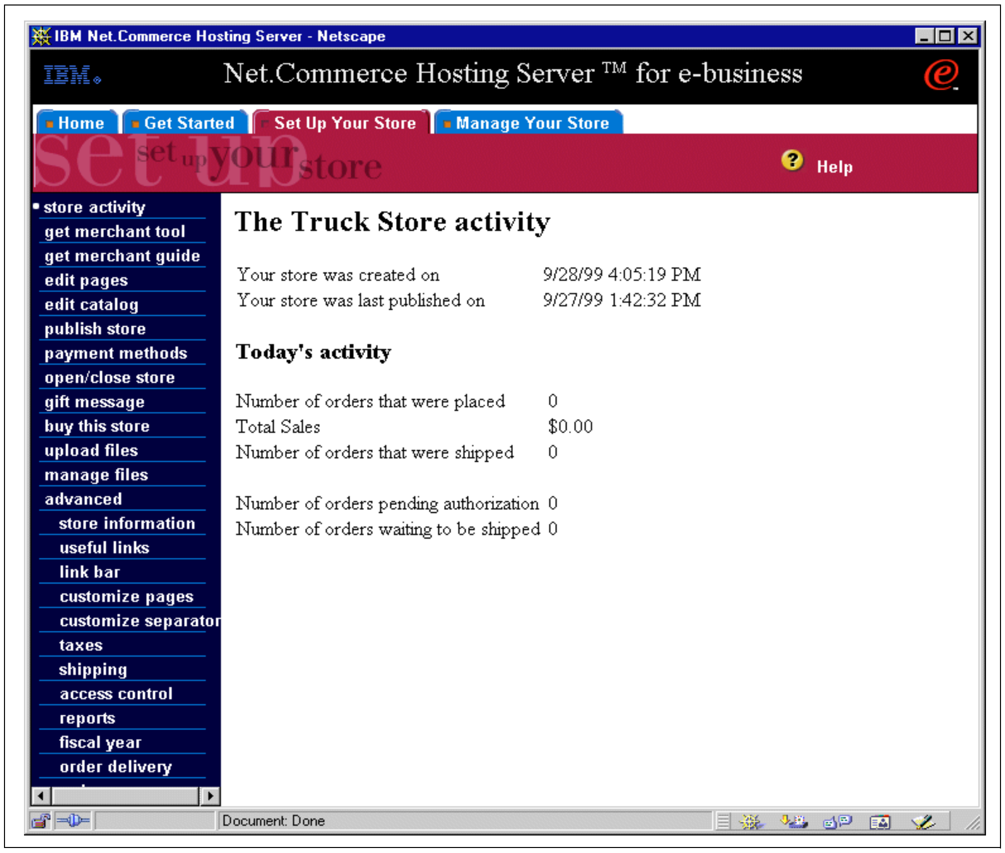
Figure 22. View Sample Stores Screen.

3.6 Customizing the process to sell a merchant store

After the merchant creates their store and publishes it, they will want to Open the store so shoppers can make purchases. At this point, the CSP can either require the merchant to purchase the store or the CSP can allow the merchant to try the store out by letting shoppers make purchases for a fixed period of time before the CSP charges the merchant for the store. Normally changing the store from New to Closed (so the merchant can Open the store) is a manual process for the CSP. However, the next two section show how to automate the process so the merchant can Open the store without any direct action from the CSP.

3.6.1 Selling a merchant store

To customize the site to allow a merchant to purchase at their store from the CSP and upon successfully completing the payment, the store will be changed from the ’New’ state to the ’Closed’ state. Thus allowing the merchant to open their store immediately.



The Truck Store activity

Your store was created on	9/28/99 4:05:19 PM
Your store was last published on	9/27/99 1:42:32 PM

Today's activity

Number of orders that were placed	0
Total Sales	$0.00
Number of orders that were shipped	0
Number of orders pending authorization	0
Number of orders waiting to be shipped	0

Figure 23. Merchant Tool with Buy this store option.

To accomplish this we must first execute the following steps:

1. Get checkout link for the product.

   Select **Manage Store** and enter the userid and password for the CHS Services Store (the default userid = ncadmin & password = ncadmin). Then enter the store name "CHS Services Store". Select the "Set Up Your Store" tab then select **edit catalog**. In the catalog editor expand Catalog; eCommerce Services; Left click on the product Online Store 1-10 Items.

Select remote content. A window will come up with three urls. Copy the "Add to Shopping Cart" url.

2. Add "Buy This Store"to Merchant Tool Menu (Set up Your Store)

   To add the new item in the menu, edit .../NetCommerce3/Tools/xml/nchs/mtool/merchantTool.xml and add the following as one line in the setUpYourStoreFolder:

   <link name="buyThisStore" url="/servlet/ShoppingCart?merchant.refno=1052&amp;product.SKU=1065" users="storeAdmin"/>

   where the value for url is the "Add to Shopping Cart" url copied in step 1. You will have to change ’&’ to ’&amp;’ as well.

3. Edit mtoolNLS.properties in .../NetCommerce3/Tools/lib/nchs.jar. Add a line:

   buyThisStore=buy this store

   in the "Merchant tool setup you store links" section. This will associate the lable ’buy this store’ with the link buyThisStore. Then replace mtoolNLS.properties file back into nchs.jar making sure the path is included. Restart the Domino Go Webserver to activate the changes.

4. Create SQL statements in either the macro order_ok.d2w or an Overridable Function that will (for the CSP store only) change the state of the merchant’s store from **New** to **Closed**.

Using a macro

Edit .../NetCommerce3/macro/en_US/<merchant_rn>/ord_ok.d2w and change:

```
%include "CSPstoremodel\ord_ok.d2w"
```

to:

```
%include "\CSPstoremodel\csp_ord_ok.d2w"
```

where <merchant_rn> stands for a merchant reference number.

Then copy

```
.../NetCommerce3/macro/common/CSPStoremodel/ord_ok.d2w
```

to

```
.../NetCommerce3/macro/common/CSPStoremodel/csp_ord_ok.d2w
```

And make the following changes to csp_ord_ok.d2w:

add to the data section:

```
store_refnum = ""
store_stat = ""
```

added functions:

```
%function(dtw_odbc) get_store_refnum()
{
       select mer_rfnbr
       from acc_usrgrp
       where usr_refnum = $(SESSION_RN)

       %REPORT {

              %ROW {
                  @dtw_assign(store_refnum, V_mer_rfnbr)
              %}
       %}
       %MESSAGE{
              100: { %} :CONTINUE
              default: { ERROR in get_store_refnum %}
       %}
%}


%function(dtw_odbc) set_store_status ()
{
   UPDATEmcspinfo
   SET mpstate = 'C'
   WHERE mpmenbr = $(store_refnum)

       %REPORT {

                  %ROW {
                  Store $(store_refnum) set to Close<p>
                  %}
       %}
       %MESSAGE{
                  100: { %} :CONTINUE
       default: { ERROR in Set_store_status %}
       %}
%}

%function(dtw_odbc) get_store_status ()
```

```
{
   SELECT mpstate
   FROM mcspinfo
   WHERE mpmenbr = $(store_refnum)

       %REPORT {

                  %ROW {
                  @dtw_assign(store_stat, V_mpstate)
                  %}
       %}
       %MESSAGE{
                  100: { %} :CONTINUE
       default: { ERROR in get_store_status %}
       %}

%}
```

added to report section:

```
@get_store_refnum()
@get_store_status()

%IF ($(store_stat) == "N")

@set_store_status()
@get_store_status()
%ENDIF
%IF ($(store_stat) == "C")
Your store is now Closed <br> Use the Merchant Tool to Open your store.<p>
%ELSE
Your store could not be placed in the Closed state. Please call you site
administrator.<p>
%ENDIF
```

Using an Overridable Function

As a better alternative we can change the status of the store in an Overridable Function (OF) that executes from the task ext_ord_proc. There are several reasoons why this would be a prefered method. First of all it is more in line with the E-Commerce framework which is the Net.Commerce v3.x internal programming model. The E.Commerce Framework prefers changes to the database be done in OFs. This is because within an OF errors a better handled and the database can be rolled back. Also related to this situation is the fact that macros can be executed directly from the browser. We do not want the merchant to execute csp_ord_ok.d2w directly without purchasing the store.

The OF OpenStore is listed in Appendix G, “Source code for OpenStore.cpp” on page 221 as the file OpenStore.cpp. This OF first of all uses the User object to get the ShopperRefNum:

```
//////////////////////////////////////////////////////////////////////////
// Get the current shopper

      User* user = (User*) Env.Seek(NC_Environment::_VAR_Shopper);
      if (user == NULL)
      {
         error.nls(&_ERR_CANT_LOAD_SHOPPER) << endl;
         return false;
      }
      String ShopperRefNum = user->getValue(User::_COL_REF_NUM);
```

We then use this ShopperRefNum to get the StoreRefNum which is the reference number of the store that is being purchased:

```
//////////////////////////////////////////////////////////////////////////
// get store ref number or store being purchased


      String StoreRefNum;
      String Stmt;

      Stmt.Clean() << "SELECT MER_RFNBR"
           << " FROM ACC_USRGRP"
           << " WHERE USR_REFNUM = " <<  ShopperRefNum;

SQL Sql2(*(DataBaseManager::GetCurrentDataBase()), Stmt);
      Row SqlRow2;
      if (Sql2.getNextRow(SqlRow2) == ERR_DB_NO_ERROR)
      StoreRefNum = SqlRow2.getCol(1);
```

Then we get the store status:

```
//////////////////////////////////////////////////////////////////////////
// get store status

      const char* MPState;
      const char*  NState = "N";

      Stmt.Clean() << "SELECT   mpstate "
           << " FROM   mcspinfo "
           << " WHERE  mpmenbr = " << StoreRefNum;
      SQL Sql(*(DataBaseManager::GetCurrentDataBase()), Stmt);
      Row SqlRow;

      if (Sql.getNextRow(SqlRow) == ERR_DB_NO_ERROR)
      {
        MPState = SqlRow.getCol(1).c_str();
      }
      else
      {
       return false;
      }
```

and if the status is ’N’ we update the status to ’C’:

```
if ( strncmp( MPState, NState, 1) == 0)
{
      String Stmt;
      Stmt << "UPDATE mcspinfo "
           << " SET  mpstate = ’C’ "
           << " WHERE  mpmenbr ="  << StoreRefNum;

      SQL Sql(*(DataBaseManager::GetCurrentDataBase()), Stmt);
}
```

which will allow the merchant to change the status to Open at will. (The SQL can be combined in this OF, however it is represented this way to illustrate the steps taken).

Compile the code using the makefile provided for your platform. The resulting file should be placed in the /NetCommerce3/bin directory.The following SQL will allow Net.Commerce to use your new OF.

for NT

```
insert into ofs (refnum , dll_name, vendor, product, name, version,
description) values
((select max(refnum) from ofs) + 1, ’OpenStore.dll’, ’IBM ITSO’, ’NC’,
’OpenStore’, 1.0 , ’Changes status of merchant store from New to Closed’);

insert into task_mer_of (task_rn, merchant_rn, of_rn) values (
(select tkrfnbr from tasks where tkname=’EXT_ORD_PROC’),1052,
(select refnum from ofs where name=’OpenStore’)
);

commit;
```

For AIX:

```
insert into ofs (refnum , dll_name, vendor, product, name, version,
description) values
((select max(refnum) from ofs) + 1, 'OpenStore.a', 'IBM ITSO', 'NC',
'OpenStore', 1.0 , 'Changes status of merchant store from New to Closed');

insert into task_mer_of (task_rn, merchant_rn, of_rn) values (
(select tkrfnbr from tasks where tkname='EXT_ORD_PROC'),1052,
(select refnum from ofs where name='OpenStore')
);

commit;
```

Replace '1052' in the above examples with the store reference number for the CSP store.After executing the above, stop and start the Net.Commerce server.

Additional information regarding Overridable Functions can be found in *"Commands, Tasks, Overridable Functions and the E-Commerce Framework"* which can be downloaded from www.ibm.com/net.commerce.

The code in the appendix for OpenStore can be compiled on either AIX or NT. The Appendix also includes the makefile for NT as the file OpenStore.nt.

5. Lastly, remove the 'buy store' menu item from the navigator frame in the main screen. Edit the file .../NetCommerce3/html/en_US/cspsite/navigation.html. Delete the following code.

```
<!-- buy store -->
           <TR><TD><IMG NAME="item9" SRC="/CHS/images/bullet_blank.gif"
WIDTH=5 HEIGHT=5 ALT="" BORDER=0></TD>
               <TD NOWRAP>&nbsp;<FONT FACE="Arial,Helvetica,sans-serif"
SIZE=2><B><A
HREF="javascript:go(9,'/cspsite/buy_store_banner.html','/cspsite/buy_store
.html')"

onMouseOver="on(9); status='buy store'; return true;" onMouseOut="off(9);
status='';">buy store</A>
                   </B></FONT></TD></TR>
           <TR><TD></TD>
               <TD><IMG SRC="/CHS/images/separator.gif" WIDTH=122 HEIGHT=1
ALT="" BORDER=0></TD></TR>

           <TR><TD></TD>
               <TD>&nbsp;</TD></TR>
```

Now the merchant can only purchase the store after creating a store. This gives the merchant the ability to open the store upon purchasing it.

3.6.2 Try and Buy

Try and Buy is a feature that would allow merchants to create a store and open for business without paying for the store. This would let the merchant try the store out on customers before purchasing the store. After an specific period of free time, the merchant must pay for the store or it would be closed by the CSP.

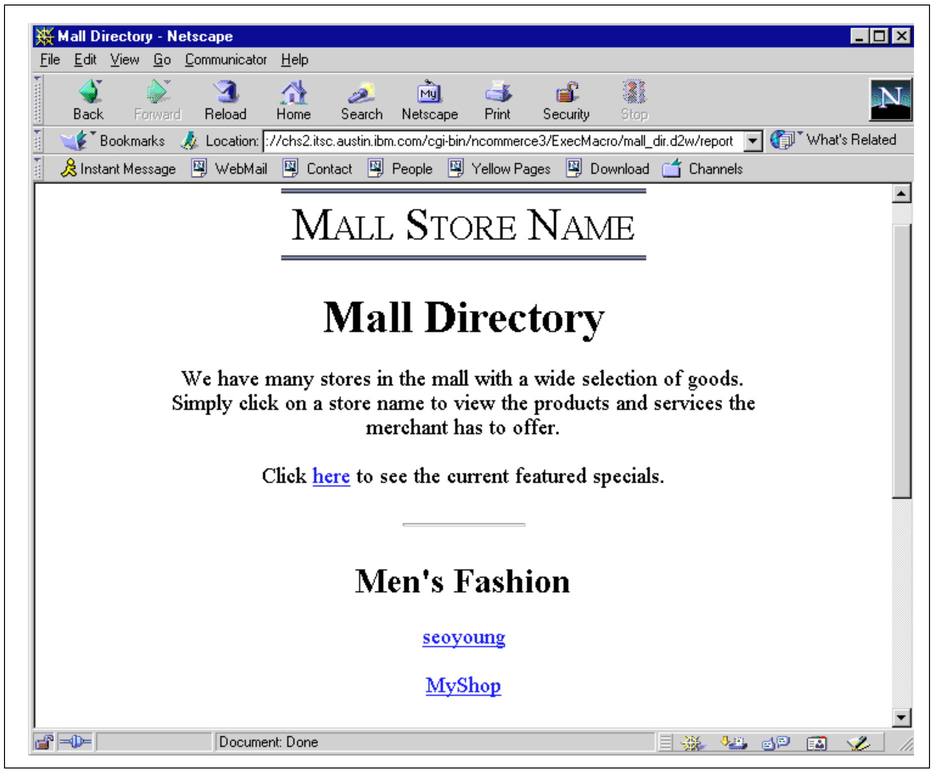
One way to accomplish this is to follow the instructions in the section 3.6.1, “Selling a merchant store” on page 57 for selling a store to a merchant, however, create the menu item as ’60 Day Trial’ and link it to the ’Add to shopping cart’ link for a product called ’60 Day Trial’ with a price of $0.00. Collect the merchant information and credit card number and let the merchant know that they will be billed at the end of 60 days if they do not cancel by e-mail.

3.7 Adding mall-wide navigation feature

Most Commerce Hosting Service Providers today provide a value added service to their customers by creating a mall to house all their merchants in. This allows the shoppers to go to one location to access multiple shops. For the CSP, it is a way to market the products and services that are offered. This section explains how a CSP can add a mall wide navigation feature to its mall.

Before going into detail on how this can be done. Figure 24 on page 65 and Figure 25 on page 66 shows the difference between a before and after a mall wide navigation feature is created.

To create a mall-wide navigation feature, a frame is built around the Mall. The example is provided based on the assumption that the user is familiar with HTML frame concepts.


Mall Directory - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: ://chs2.itsc.austin.ibm.com/cgi-bin/ncommerce3/ExecMacro/mall_dir.d2w/report
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
Document: Done

MALL STORE NAME

Mall Directory

We have many stores in the mall with a wide selection of goods. Simply click on a store name to view the products and services the merchant has to offer.

Click here to see the current featured specials.

Men's Fashion

seoyoung

MyShop

Figure 24. Before mall wide navigartion feature is created

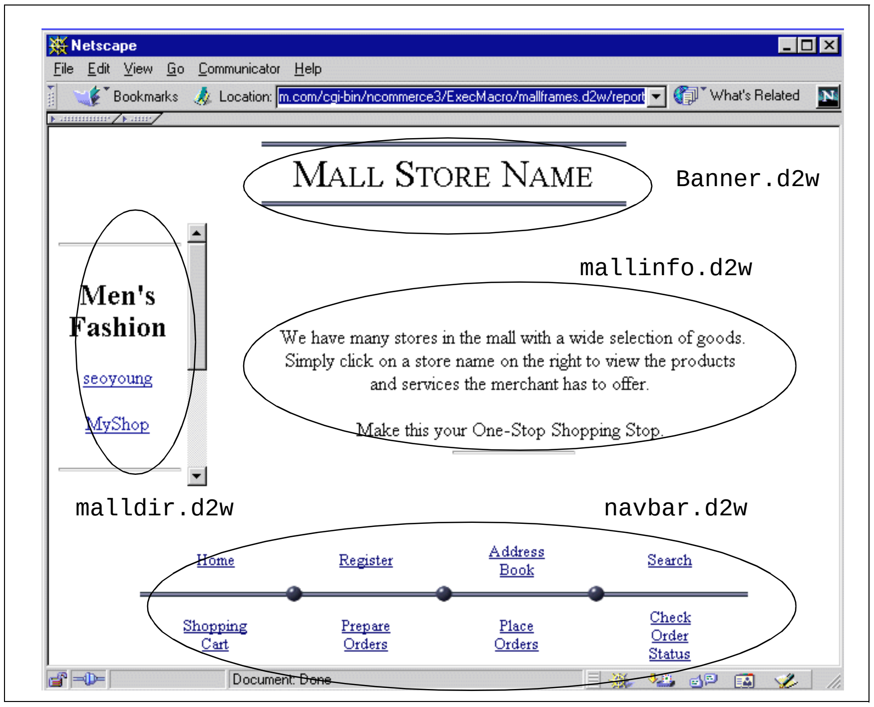

Netscape
File Edit View Go Communicator Help
Bookmarks
Location: m.com/cgi-bin/ncommerce3/ExecMacro/mallframes.d2w/report
What's Related
MALL STORE NAME
Banner.d2w
mallinfo.d2w
Men's Fashion
seoyoung
MyShop
We have many stores in the mall with a wide selection of goods. Simply click on a store name on the right to view the products and services the merchant has to offer.
Make this your One-Stop Shopping Stop.
malldir.d2w
navbar.d2w
Home
Register
Address Book
Search
Shopping Cart
Prepare Orders
Place Orders
Check Order Status
Document: Done

Figure 25. Mall Wide Navigation Feature added

On the left of the window in Figure 25 on page 66, there is a navigation bar which allows the shoppers to select the stores in which they are interested. When a store is selected, the contents of the store is displayed in the main window without changing the rest of the features in the page as shown in Figure 26 on page 67. In this example "MyShop" is selected and the store is displayed in the main window. Notice that the Mall identity is still preserved and the Mall directory can still be seen on the left window.

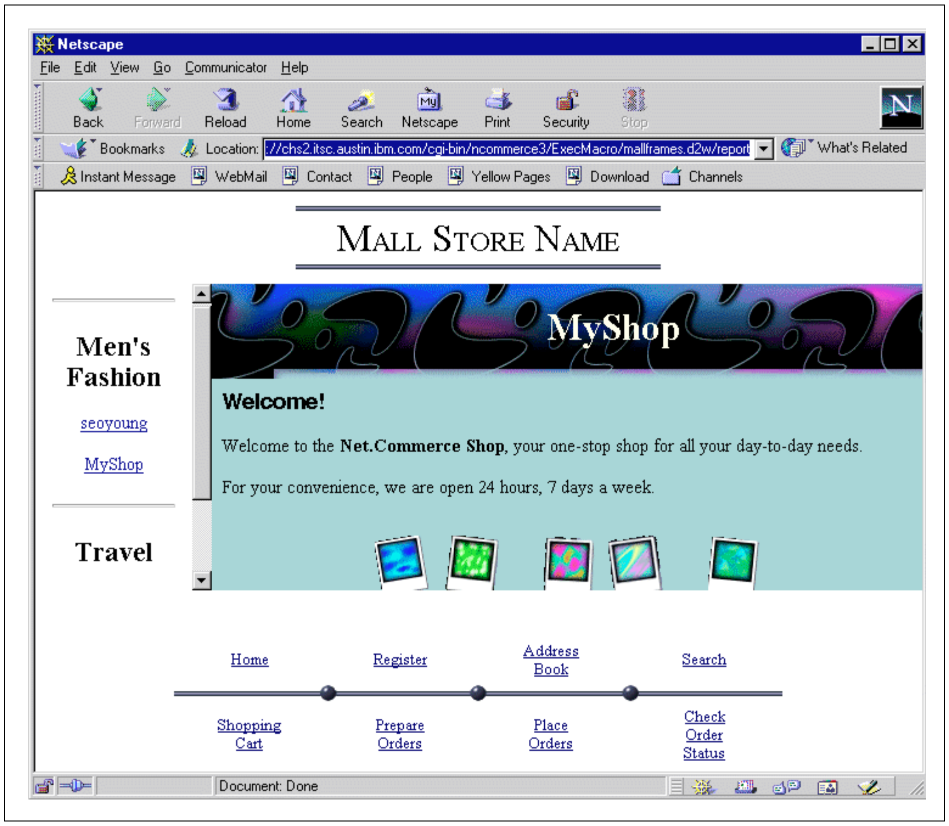

Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: //chs2.itsc.austin.ibm.com/cgi-bin/ncommerce3/ExecMacro/mallframes.d2w/report
What's Related
Instant Message WebMail Contact People Yellow Pages Download Channels
MALL STORE NAME
Men's Fashion
seoyoung
MyShop
Travel
MyShop
Welcome!
Welcome to the Net.Commerce Shop, your one-stop shop for all your day-to-day needs.
For your convenience, we are open 24 hours, 7 days a week.
Home Register Address Book Search
Shopping Cart Prepare Orders Place Orders Check Order Status
Document: Done

Figure 26. The Mall page with a shop selected

To create a mall wide navigation feature, information regarding the links to the stores are required. As an example, we will modify the Demomall that is provided as part of NCHS. The macros and html files of the Demomall can be located in the following directories respectively:

```
/usr/lpp/NetCommerce3/macro/en_US/demomall
/usr/lpp/NetCommerce3/html/en_US/demoamall
```

When accessing the Demomall home page,

```
http://<hostname>/demomall/basemall.htm
```

the following macro mall_dir.d2w is executed when the "Guest Shopper" link is selected.This macro retrieves information about the Demomall and the stores in the Demomall from the database and displays the information on the page. We can directly execute the macro with this command on the browser:

```
http://<hostname>/cgi-bin/ncommerce3/ExecMacro/mall_dir.d2w/report
```

We will use the macro mall_dir.d2w as the basis of our example. Using TABLES and FRAMES, and knowing which macro to modify, adding a mall wide navigation frame can easily be accomplished.

1. Create a new macro which divides the page into 4 frames. An example of the macro is listed:

```
mallframes.d2w

%define {
  SHOWSQL="NO"
%}


%HTML_REPORT {

<HTML>

<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">

</HEAD>

<FRAMESET ROWS="80,*,150" BORDER="0" FRAMEBORDER="NO">

  <FRAME NAME="Banner"
SRC="/cgi-bin/ncommerce3/ExecMacro/demomall/Banner.d2w/report"
FRAMEBORDER="NO" SCROLLING="NO">

  <FRAMESET COLS="20%, *">
    <FRAME NAME="Sidenav"
SRC="/cgi-bin/ncommerce3/ExecMacro/demomall/malldir.d2w/report"
FRAMEBORDER="0">
    <FRAME NAME="Main"
SRC="/cgi-bin/ncommerce/ExecMacro/demomall/mallinfo.d2w/report"
FRAMEBORDER="0">
  </FRAMESET>

  <FRAME NAME="Navbar"
SRC="/cgi-bin/ncommerce3/ExecMacro/demomall/navbar.d2w/report"
FRAMEBORDER="NO" SCROLLING="NO">
</FRAMESET>
</HTML>

%}
```

The above macro creates a page with 4 frames. Each frame will then execute its own macro again. A brief description of each macro is provided:

- banner.d2w - Displays mall information.
- malldir.d2w - Displays the list of stores in the mall.
- mallinfo.d2w - Displays information about the mall.
- navbar.d2w - Displays the navigation bar for the mall.

2. The code for the macros above are actually all taken from mall_dir.d2w and put into separate macro files so that each macro is executed in a separate frame. For illustration purposes, we will create malldir.d2w as an example. Use the main function from mall_dir.d2w and apply it in the new macro malldir.d2w. Do the same for the rest of the macros, using the relevant parts.The main function used in malldir.d2w is shown below: (Please refer to the Appendix A for the full code)

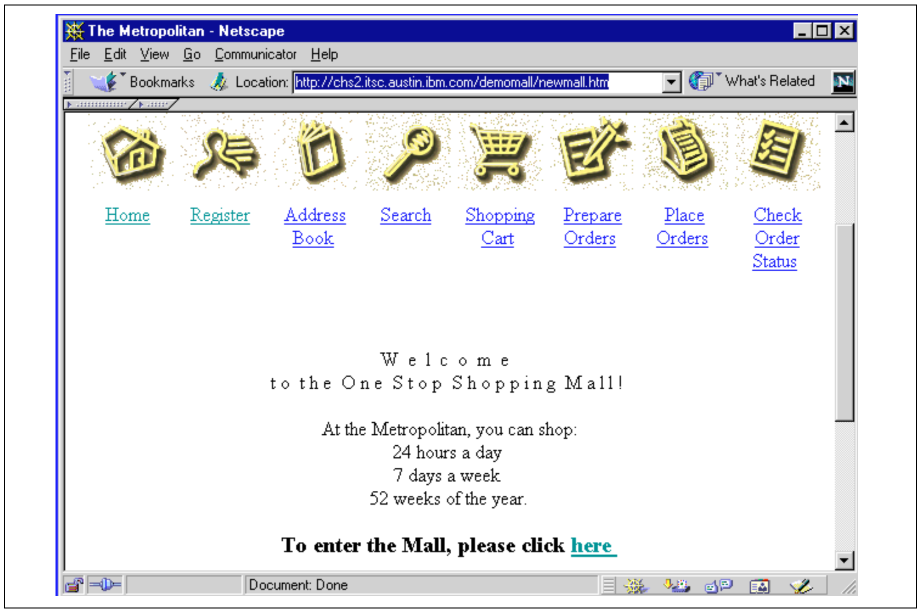
```
%function(dtw_odbc) mall_dir(){

    select merfnbr, mestname, mescnbr, scgry, metbase, methmb
    from merchant, strcgry
    where mescnbr=scrfnbr
    order by scgry

  %REPORT{
   %ROW{

      %if ((V_mescnbr == last_prod) && (V_methmb != null))
       <HR width=100> <BR>
       <img src="$(V_methmb)"><BR>
        <A HREF="$(V_metbase)" Target="Main">$(V_mestname)</A><P>
      %elif ((V_mescnbr == last_prod) && (V_methmb == null))
         <A HREF="$(V_metbase)" Target="Main">$(V_mestname)</A><P>
      %elif ((V_mescnbr != last_prod) && (V_methmb != null))
        <HR width=100> <h2>$(V_scgry)</h2> <img src="$(V_methmb)"><BR>
        <A HREF="$(V_metbase)" Target="Main">$(V_mestname)</A><P>
      %else
        <HR width=100> <h2>$(V_scgry)</h2>
        <A HREF="$(V_metbase)" Target="Main">$(V_mestname)</A><P>
      %endif
  %}
%}
```

3. Once the 4 macros are created, place them in the macro directory.

```
/usr/lpp/NetCommerce3/macro/en_US/demomall
```

4. Create an HTML file that will execute the first macro created in Step 1 (See Figure 27 on page 70). A very simple command that can be used in the HTML file would be:

```
<A HREF="/cgi-bin/ncommerce3/ExecMacro/<macro file name from Step
1>/report">Click here to Enter the Mall</A>.
```


The Metropolitan - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://chs2.itsc.austin.ibm.com/demomall/newmall.htm
What's Related
Home
Register
Address Book
Search
Shopping Cart
Prepare Orders
Place Orders
Check Order Status
Document: Done

W e l c o m e
to the One Stop Shopping Mall!

At the Metropolitan, you can shop:
24 hours a day
7 days a week
52 weeks of the year.

To enter the Mall, please click here

Figure 27. A Mall Page

You may want to take the user directly to the Mall directory instead of having them selecting to enter the mall by clicking on a link. One way of doing so would be to code all the frames logic into a HTML file. The code to do so is listed below:

```
<HTML>
```

```
<HEAD>

<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>$(LongStoreName)</TITLE>

</HEAD>


<FRAMESET ROWS="80,*,150" BORDER="0" FRAMEBORDER="NO">

  <FRAME NAME="Banner"
SRC="/cgi-bin/ncommerce3/ExecMacro/demomall/Banner.d2w/report"
FRAMEBORDER="NO" SCROLLING="NO">

  <FRAMESET COLS="20%, *">
    <FRAME NAME="Sidenav"
SRC="/cgi-bin/ncommerce3/ExecMacro/demomall/malldir.d2w/report"
FRAMEBORDER="0">
    <FRAME NAME="Main"
SRC="/cgi-bin/ncommerce/ExecMacro/demomall/mallinfo.d2w/report"
FRAMEBORDER="0">
  </FRAMESET>

  <FRAME NAME="Navbar"
SRC="/cgi-bin/ncommerce3/ExecMacro/demomall/navbar.d2w/report"
FRAMEBORDER="NO" SCROLLING="NO">
</FRAMESET>

</HTML>
```

After saving the code above to a HTML file (e.g. onestopmall.htm) in the directory

```
/usr/lpp/NetCommerce3/html/en_US/demomall
```

access the site from the browser using the following URL:

```
http://<hostname>/demomall/onestopmall.htm
```

Once that is done, you are all set and ready. The mall wide navigation feature is ready to be used. Please note that the store macro files have to be modified such that the contents are displayed in the correct frames.

3.8 Different levels of service by each merchant

As a Commerce Service Provider (CSP), you might want to place restrictions on your merchants. Different fees can be charged for different levels of services.

In this section we will add a new feature (e.g. gift message feature, etc.) that is accessible by merchants who are authorized to do so. To implement this verification, we will use the field **SHFIELD1** which is reserved for merchant customization in the **SHOPPER** table. The **SHFIELD1** can be set to either

- "0" - the merchant is not authorized to access the new feature or
- "1" - the merchant is authorized to access the new feature.

Each time a merchant logs in and selects the new feature, the field **SHFIELD1** is checked. If the merchant has authorization, they will be allowed to access that page. If the merchant does not have authorization, they are given the option to purchase that access from the CSP.

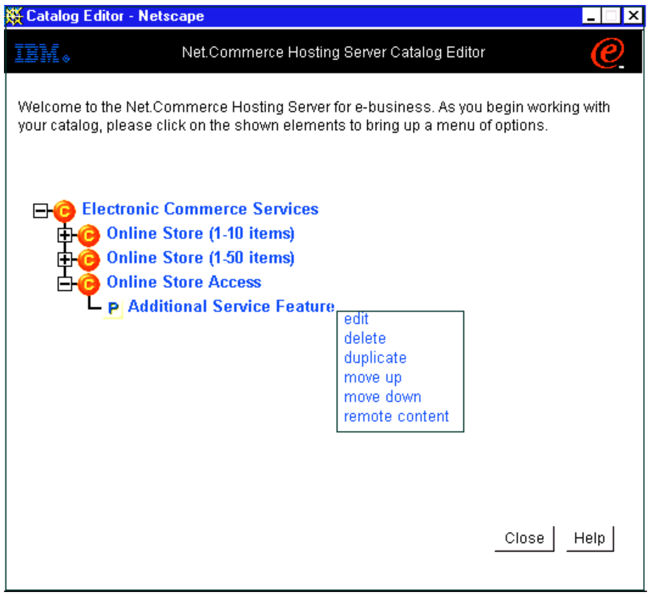
Once the merchant has purchased the access, the **SHFIELD1** is updated to "1" such that the merchant can access the new feature in subsequent logons.

To do this, there are several pieces of information that you need to obtain first.

1. Merchant reference number and the product SKU

One way of allowing your merchants to have additional features in their shops would be to allow them to purchase that access from your CHS Service Store. The product (Additional Service Feature is a product in the CHS Service Store) must first be inserted into the catalog. Once that is done, the information regarding the CHS Service Store and the product can be retrieved. An easy way to obtain all this information is by accessing the CHS Service Store and editing the products in the store.

- Log in as the CHS Service Store Administrator (use the sample provided or create your own service store) and select **edit catalogs** from the menu item under the **Set Up Your Store** tab.
- Select the product (in this example, the product selected will be the Additional Service Feature item)
- Move the cursor over the product item, and left click the mouse. A new menu will appear on the page as shown in Figure 28 on page 73.


Catalog Editor - Netscape

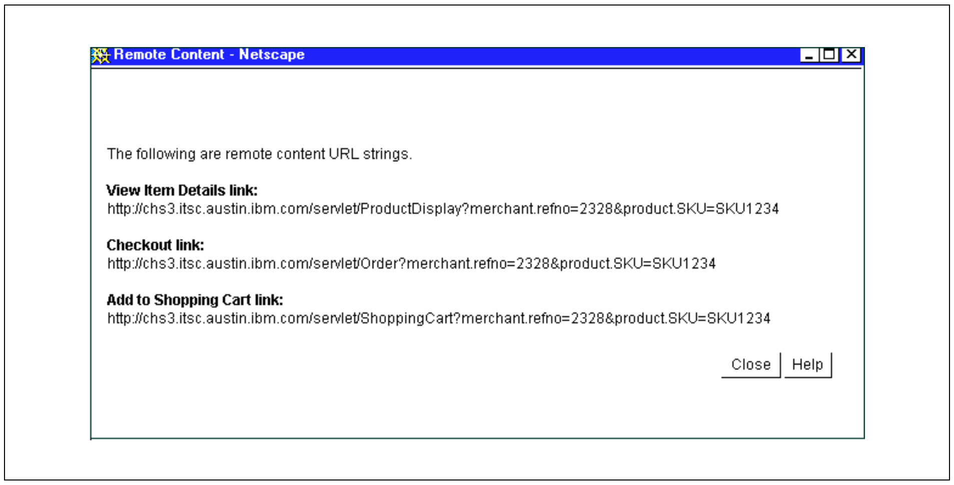
Net.Commerce Hosting Server Catalog Editor

Welcome to the Net.Commerce Hosting Server for e-business. As you begin working with your catalog, please click on the shown elements to bring up a menu of options.

Electronic Commerce Services
Online Store (1-10 items)
Online Store (1-50 items)
Online Store Access
Additional Service Feature
edit
delete
duplicate
move up
move down
remote content
Close
Help

Figure 28. Service Store

- Select the option **Remote Content** to obtain the following information shown in Figure 29 on page 74.



The following are remote content URL strings.

View Item Details link:
http://chs3.itsc.austin.ibm.com/servlet/ProductDisplay?merchant.refno=2328&product.SKU=SKU1234

Checkout link:
http://chs3.itsc.austin.ibm.com/servlet/Order?merchant.refno=2328&product.SKU=SKU1234

Add to Shopping Cart link:
http://chs3.itsc.austin.ibm.com/servlet/ShoppingCart?merchant.refno=2328&product.SKU=SKU1234

Figure 29. Result of selecting Remote Content

Under the heading **Add to shopping cart link**, the information regarding the command, the merchant reference number and the product SKU is listed.

2. Add a menu item in the merchant tool.

In this example, we will call this new feature, **Additional service** and we will place this new menu item just after the **open/close Store** menu item in the merchant tool, under the **Set Up Your Store** tab. Note that this can be replaced by any name and be placed anywhere in the merchant tool. Open the file

`/usr/lpp/NetCommerce3/Tools/xml/nchs/merchantTool.xml`

In this file locate the following code:

```
<link name = "openCloseStore"
url   = "/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.StoreState"
users = "storeAdmin" />
```

Add the following code after the **openCloseStore** section:

```
<link name  = "openCloseStore"
url   = "/servlet/MerchantAdmin?DISPLAY=CTnchs.mtool.StoreState"
users = "storeAdmin" />
<link name  = "AdditionalService"
url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/service.d2w/report?mer
fnb=$env.merchant_id$"
```

```
users = "storeAdmin,catalogAdmin" />
```

Save and close the file.

A corresponding change needs to be done in mToolsNLS.properties file located in the nchs.jar file. The nchs.jar file is similar to a zipped file and can be open using applications like WinZip. Open the file:

```
/usr/lpp/NetCommerce3/Tools/lib/nchs.jar
```

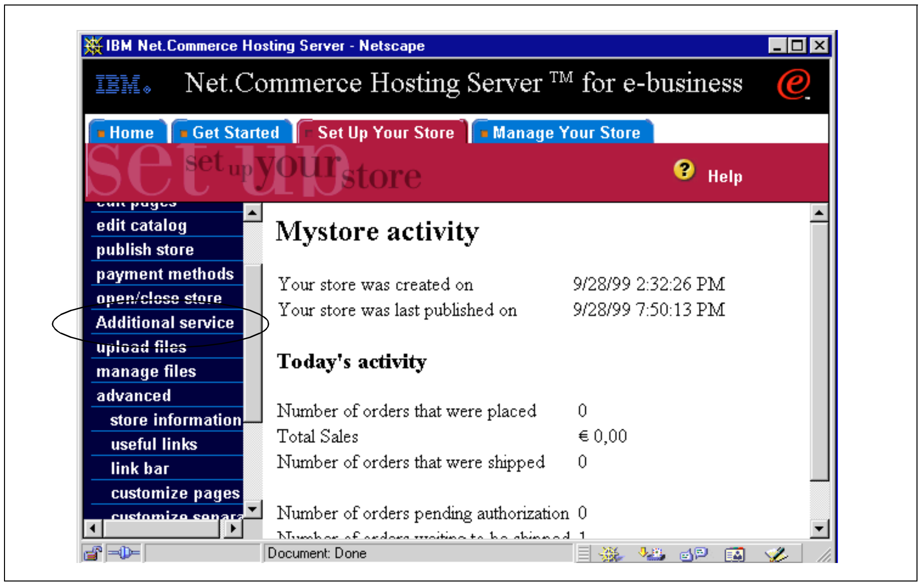
Extract and open the file

```
mtoolsNLS.properties
```

Locate the section called **# Merchant tool set up your store links** and add the following AdditionalService link to that section:

```
# Merchant tool set up your store links
AdditionalService= Additional service
storeActivity=store activity
getMerchantTool=get merchant tool
getMerchantGuide=get merchant guide
publishStore=publish store
paymentMethods=payment methods
editPages=edit pages
editCatalog=edit catalog
```

Save and close the file. If you access the merchant tool now, you should see the new menu item **Additional service** located just after the **open/close Store** as shown in Figure 30 on page 76.



Mystore activity

Your store was created on	9/28/99 2:32:26 PM
Your store was last published on	9/28/99 7:50:13 PM

Today's activity

Number of orders that were placed	0
Total Sales	€ 0,00
Number of orders that were shipped	0
Number of orders pending authorization	0
Number of orders waiting to be shipped	1

Figure 30. Additional service menu item is added

3. Selecting **Additional service**

When a new menu item is added to the merchant tool, a corresponding code must be created and then executed when the new menu item is selected. In **Step 1** we created a new menu in the merchant tool. Now we will create the action (code) that will be executed when the **Additional service** menu item is selected.

We inserted the following code in the merchantTool.xml file:

```
<link name  = "AdditionalService"
url="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/service.d2w/report?mer
fnb=$env.merchant_id$"
users = "storeAdmin,catalogAdmin" />
```

When the Additional service menu item is selected, the macro service.d2w is executed with the merchant reference number as the parameter.

The macro service.d2w will first check if the merchant has authority to access the **Additional service** feature. The code to check the field **SHFIELD1** in the **SHOPPER** table is listed below:

```
%function(dtw_odbc) check_status()
{
   SELECT shfield1, shlogid
   FROM shopper
   WHERE shlogid='$(SESSION_ID)'

   %REPORT {
    %ROW {
     @DTW_assign(STATUSFIELD, V_shfield1)
   %}
    %}
```

The function above selects the **SHFIELD1** from the **SHOPPER** table and then assigns the result to the variable **STATUSFIELD**.

In the **HTML Report Section** of the macro, we will check the STATUSFIELD and then point the merchant to the right link.

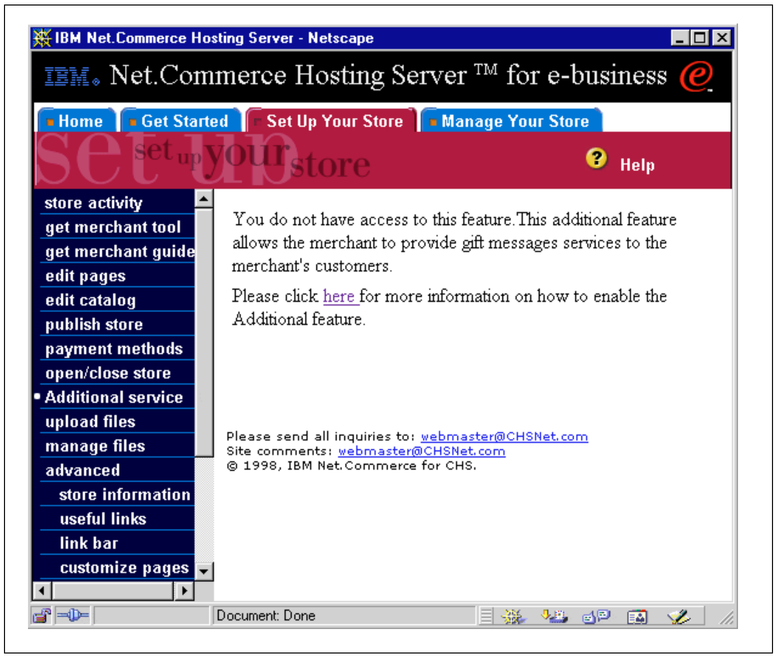
```
@check_status()

%if (STATUSFIELD != "1")

  <TABLE>
    <TR>
      <TD>You do not have access to this feature. This additional feature
allows the merchant to provide gift messages service to your customers.
</TD>
    </TR>
    <TR>
      <TD>Please click <A
href="/servlet/ShoppingCart?merchant.refno=2328&amp;product.SKU=SKU1234">h
ere </A> for more information on how to enable the Additional feature.</TD>
    </TR>
  </TABLE>

%else
  <TABLE>
    <TR>
      <TD>YES! Congratulations you now have access to this feature.</TD>
    </TR>
  </TABLE>

%endif
```

Save the macro service.d2w in directory

```
/usr/lpp/NetCommerce2/macro/en_US/ncadmin/storemgr
```

Please refer to the Appendix for the full sample code.

When the merchant clicks on the **Additional service** link, the following is displayed on the browser as shown in Figure 31 on page 78.



You do not have access to this feature. This additional feature allows the merchant to provide gift messages services to the merchant's customers.

Please click here for more information on how to enable the Additional feature.

Please send all inquiries to: webmaster@CHSNet.com
Site comments: webmaster@CHSNet.com
© 1998, IBM Net.Commerce for CHS.

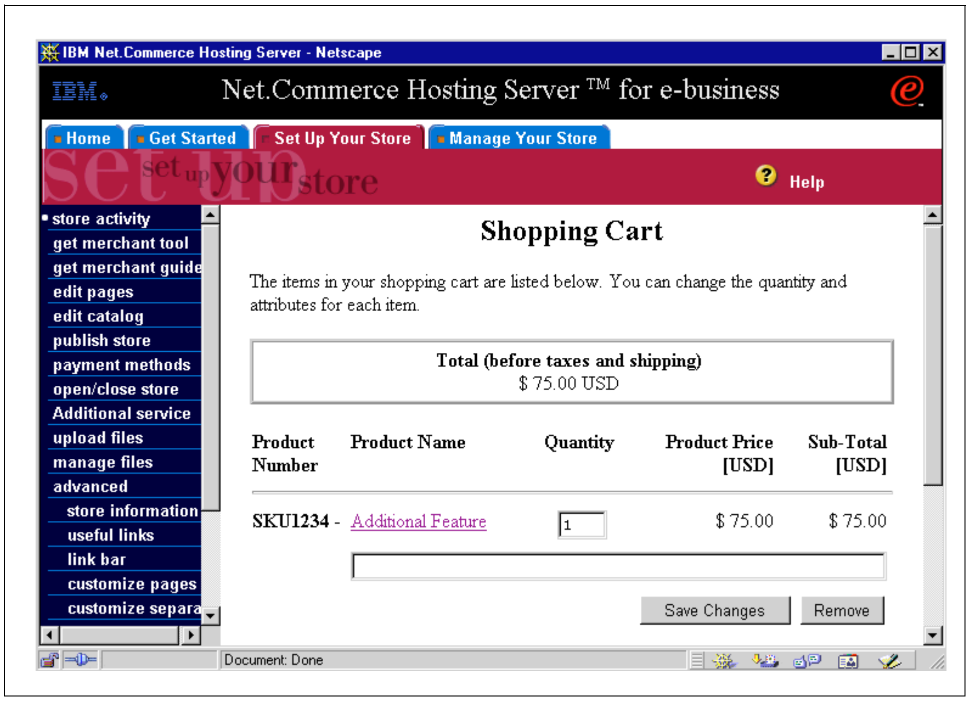
Figure 31. When the merchant with no authorization selects the Additional service link

4. Linking to the shopping cart.

If the merchant chooses to purchase the authorization to access the Additional feature, the merchant will be brought to the CSP’s Service Store’s shopping cart page with the product (Additional Service Feature is sold as a product in the CSP’s Service Store) already placed in the shopping cart as shown in Figure 32 on page 79. The code to link to the shopping cart page is listed below:

```
Please click <A
href="/servlet/ShoppingCart?merchant.refno=2328&amp;product.SKU=SKU1234">h
ere </A> for more information on how to enable the Additional feature.
```

The command, merchant reference number and the product SKU is obtained from Step 1.



Shopping Cart

The items in your shopping cart are listed below. You can change the quantity and attributes for each item.

Total (before taxes and shipping)
$ 75.00 USD

Product Number	Product Name	Quantity	Product Price [USD]	Sub-Total [USD]
SKU1234 -	Additional Feature	1	$ 75.00	$ 75.00

Figure 32. The Shopping Cart page when the merchant chooses to purchase access

5. Updating the **SHFIELD1** field in the **SHOPPER** table

Once the merchant has purchased the access, remember that the **SHFIELD1** in the **SHOPPER** table has to be updated to reflect the changes. This update should be executed in the macro ord_ok.d2w in the CSP’s service store.

- Edit the file

```
/usr/lpp/NetCommerce3/macro/<merchant reference number of CHS Service
Store>/ord_ok.d2w
```

The content of the file will show

```
%include "<merchant reference number of CHS Service Store>/include.inc"
%include "CSPstoremodel/ord_ok.d2w"
```

- Change the second include statement to

```
%include "<merchant reference number of CHS Service Store>/ord_ok.d2w"
```

- Copy ord_ok.d2w from the CSPstoremodel directory and rename it to new_ord_ok.d2w in the directory

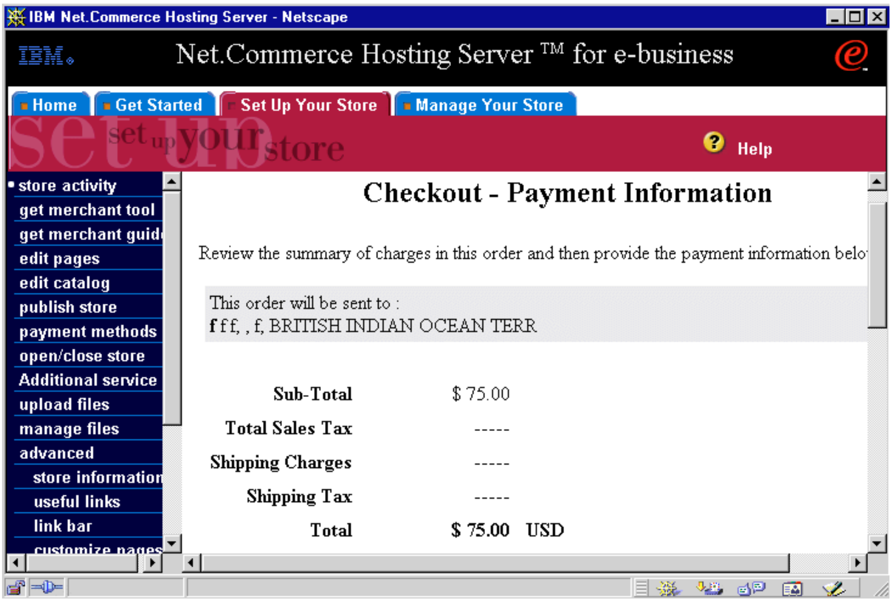
```
/usr/lpp/NetCommerce3/macro/<merchant reference number of CHS Service
Store>/
```

Any references made to ord_ok.d2w will now point to the new_ord_ok.d2w.

The macro page on the browser is shown in Figure 33 on page 80. The code to update the **SHOPPER** table is :

```
UPDATE    shopper
 SET       shfield1 = '1'
 WHERE     shlogid = '$(SESSION_ID)'
```

The above code is placed in the ord_ok.d2w macro because that is the macro where the merchant will confirm the order and the payment. Please refer to the Appendix for the full sample code.



Checkout - Payment Information

Review the summary of charges in this order and then provide the payment information belo

This order will be sent to :
f f f, , f, BRITISH INDIAN OCEAN TERR

Sub-Total	$ 75.00
Total Sales Tax	-----
Shipping Charges	-----
Shipping Tax	-----
Total	$ 75.00 USD

Figure 33. Payment Information and Checkout page (ord_ok.d2w)

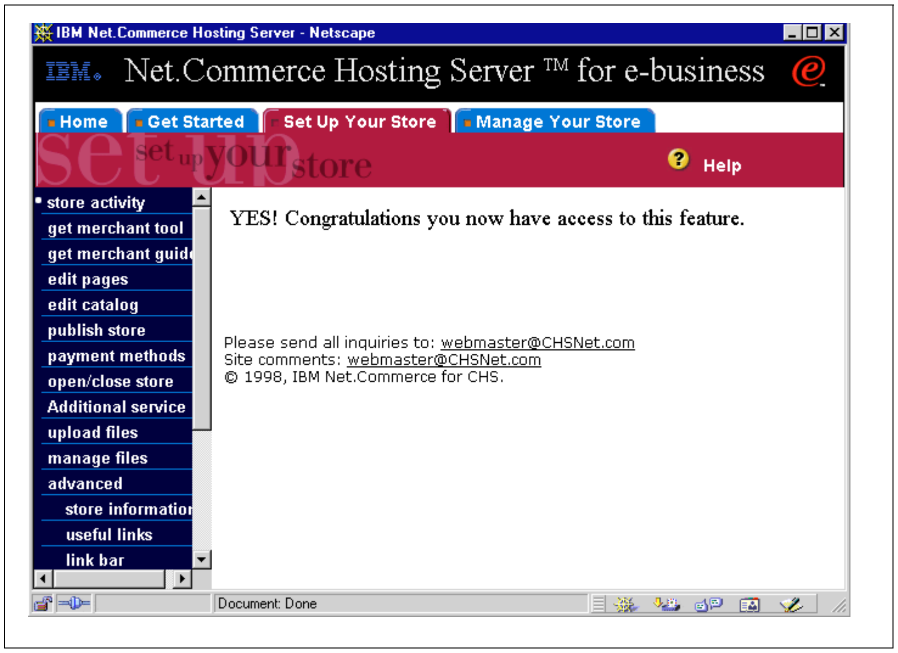
Once the ord_ok.d2w macro has been modified, you can now use the restriction feature to allow different levels of services for your merchants.

Save the ord_ok.d2w in directory

```
/usr/lpp/NetCommerce3/macro/en_US/<merchant reference number of your the
CHS Service Store>
```

If the merchant now selects **Additional service** menu item, they will have access to that page as seen in Figure 34 on page 81. In this example, when the merchant has access, a page informing the merchant of their success is used.

```
<TABLE>
    <TR>
      <TD>YES! Congratulations you now have access to this feature.</TD>
    </TR>
  </TABLE>
```



YES! Congratulations you now have access to this feature.

Please send all inquiries to: webmaster@CHSNet.com
Site comments: webmaster@CHSNet.com
© 1998, IBM Net.Commerce for CHS.

Figure 34. You now have access.

6. Making sure only purchases of the Additional Feature product will update the **SHFIELD1**.

As a CSP, your Service Store would have a variety of products and services for sale, the Additional Service Access being one of the products. In Step 5, we updated the **SHFIELD1** in the macro ord_ok.d2w. This macro is used everytime a purchase a made, regardless of the products selected. To ensure that **SHFIELD1** is only updated when the Additional Feature is selected, some additional modifications are required.

In the macro ord_ok.d2w, look for

```
%function(dtw_odbc) DISPLAY_DETAILS_LIST()
```

In the SELECT statement, add the field **PRNBR** in. This will select **PRNBR** (the SKU number) from the database.

```
select strfnbr, stsanbr, stshnbr, stmenbr, stprnbr, stprice, stquant,
stcpcur, prrfnbr, prldesc2, prsdesc, salname, safname, prnbr
from shipto, product, shaddr
wherestshnbr=$(SESSION_RN) and stmenbr=$(MerchantRefNum) and
stprnbr=prrfnbr and stornbr=$(order_rn)
and stsanbr=sarfnbr
order by stmenbr, stsanbr, strfnbr
```

Once that is done, insert the following logic to check the **PRNBR** under the same function, under the %ROW section:

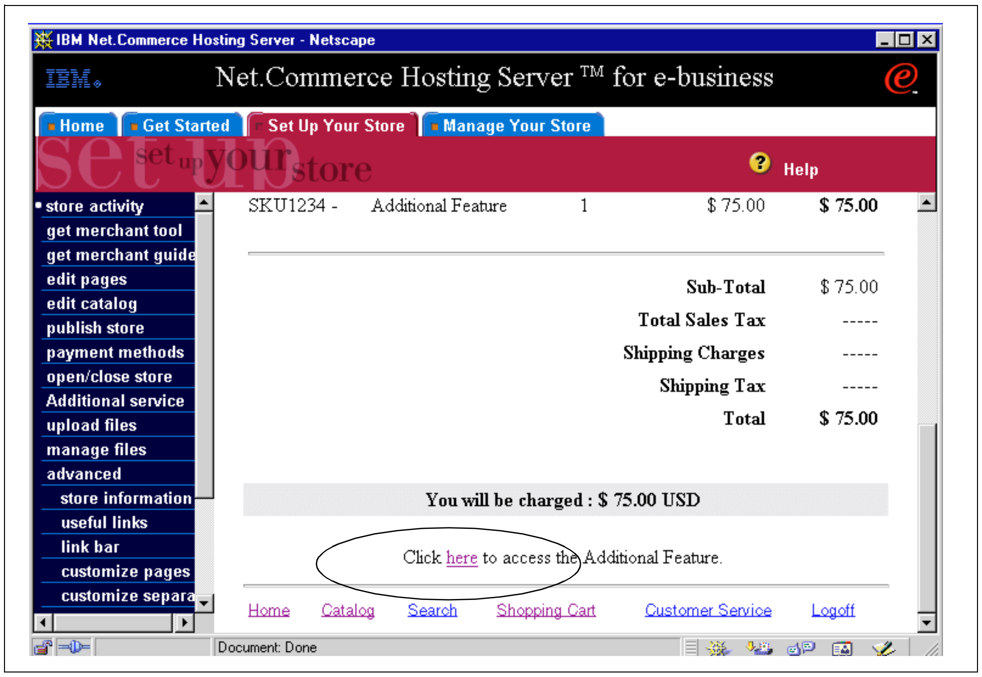
```
%ROW{

@DTW_ASSIGN(PRODSKU, V_prnbr)

%IF (PRODSKU == "SKU1234")
@DTW_ASSIGN(UPDATE, "1")
%ENDIF
```

From Step 1, we know that the Additional Feature SKU is **SKU1234**, we will use that SKU number in the **if statement**.

In the HTML Report Section, place the following code just before **@DISPLAY_CUSTOM_NAVBAR()**

```
%IF (UPDATE == "1")
@update_status() #code to update the SHFIELD1
<center>
 <TABLE>
   <TR>
    <TD>Click <A
href="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/service.d2w/report?me
rfnb=$(MerchantRefNumber)">here</A> to access the Additional Feature. </TD>
   </TR>
 </TABLE>
</center>
%ENDIF
```

The code above will only update the **SHFIELD1** only if the Addtional Feature product is chosen and allow the user to link to the Additional Service page from the Order Confirmation page as shown in Figure 35 on page 83.



SKU1234 -	Additional Feature	1	$ 75.00	**$ 75.00**

Sub-Total	$ 75.00
Total Sales Tax	-----
Shipping Charges	-----
Shipping Tax	-----
Total	**$ 75.00**

You will be charged : $ 75.00 USD

Click here to access the Additional Feature.

Figure 35. After a successful Purchase

3.9 Changing store creation process

3.9.1 Introduction

During the store creation process, not much information about the merchant is gathered. As designed, Net.Commerce Hosting Server requires a logon ID, password, store name and store currency in order to create a store. Once a store has been created, a merchant can, but not necessarily, enter additional information about the store such as contact name, address, phone and store description by clicking the Store Information button under the **Set Up Your Store** tab of the merchant tool.

The default process is as follows. When a merchant clicks **Create Store** from the CSP site, the **Create Profile** form is presented. After completing this form, the store is created and a merchant can then, (if they choose to do so), fill in additional information about themselves and the store by choosing **Store Information**

In some cases the CSP may want to present the **Store Information** form to the merchant as part of the initial step of store creation in order to gather more information about the merchant and the store being created before store creation has actually occurred. This ensures the maximum amount of information is gathered regarding the store.

In order to present the **Store Information** form in the initial step of store creation, the **Create Profile** form can be modified to present the **Store Information** form after it is submitted rather than calling the store creation command. The **Store Information** form can then call the store creation command. Essentially, a copy of the **Store Information** form is inserted into the store creation flow. After adding this customization, if a merchant chooses **Store Information** from the **Set Up Your Store** tab after store creation, the functionality will remain the same. Figure 36 illustrates the store creation flow both before and after customization.

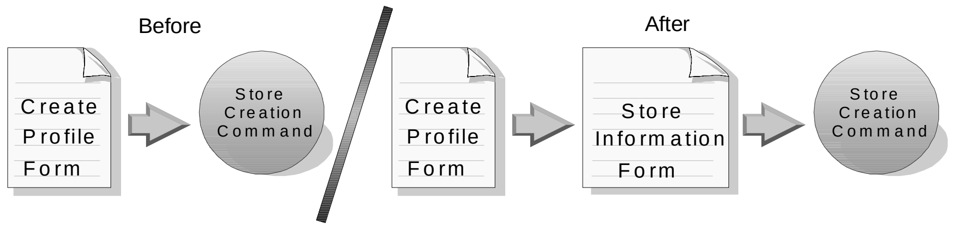


Figure 36. Customizing store creation flow

3.9.2 Changing store creation process

The following sections will guide you in implementing this solution on AIX. A similar guide for Windows NT can be found on Appendix G.1, “Customizing store creation process on Windows NT” on page 181.

1. Make a copy of the **Store Information** form to be inserted into the store creation flow.

   The **Store Information** form layout is contained in the StoreInfo.tem file. This file defines how the form looks, what text is used and the actions the form performs. A copy of this file will be modified and inserted into the store creation flow.

Log in as root and make a copy of the StoreInfo.tem file located in the /usr/lpp/NetCommerce3/Tools/mpg_templates/nchs/mtool/ directory. Name this copy Register2.tem and make sure its permissions are set correctly by issuing the following command:

```
> chmod 666 Register2.tem
```

Note

Make sure that the file being modified in step 2 is Register.tem and the file being modified in step 3 is Register2.tem.

2. Modify the **Store Creation** form to present the new **Store Information** form after it is submitted.

   The **Store Creation** form layout is contained in the Register.tem file. This file defines how the form looks, what text is used and the actions the form performs. This file will be modified to call the new **Store Information** form instead of the store creation command.

   To edit the Register.tem file, add write permission to the file.

   ```
   > chmod +w Register.tem
   ```

   Open the file Register.tem in a text editor and comment out the following line, by adding -- to the beginning of each line. This will disable the call to the store creation command by this form.

   ```
   -- /*
   --<SCRIPT>top.location.href='http://$env.hostname$/servlet/MerchantAdmi
   n?PROCESS=CTnchs.mtool.Filter&XMLFile=nchs.mtool.merchantTool.xml&start
   ingFolder=getStartedFolder'; </SCRIPT>
   -- */
   ```

   To direct this form to the new **Store Information** form, add the following line just above the commented out line:

   ```
   /*
   <SCRIPT>location.href='http://$env.hostname$/servlet/MerchantAdmin?DISP
   LAY=CTnchs.mtool.Register2'; </SCRIPT>
   */
   ```

Save this file and exit. Change the permissions back to their original state.

```
> chmod 666 Register.tem
```

3. Modify the new **Store Information** form to call the store creation command.

   To edit the Register2.tem file, add write permission to the file.

   ```
   > chmod +w Register2.tem
   ```

   Open the file Register2.tem in a text editor and comment out the following line, by adding -- to the beginning of each line. This will disable the displaying of the merchant tool **Get Started** tab.

   ```
   -- /*
   --<SCRIPT>window.location="http://$env.hostname$/servlet/MerchantAdmin?
   DISPLAY=CTnchs.mtool.StoreInfoConfirm"; </SCRIPT>
   -- */
   ```

   In order to direct the new **Store Information** form to call the store creation command, add the following line after the commented out line:

   ```
   /*
   <SCRIPT>top.location.href='http://$env.hostname$/servlet/MerchantAdmin?
   PROCESS=CTnchs.mtool.Filter&XMLFile=nchs.mtool.merchantTool.xml&startin
   gFolder=getStartedFolder'; </SCRIPT>
   */
   ```

   To display the correct error messages during the new step in the store creation process, find the line containing the `$mtoolNLS.storeInfoErrorMandatoryTop$` variable and change it to `$mtoolNLS.errorMandatoryTop$`, then find the line containing the `$mtoolNLS.storeInfoErrorMandatoryEnd$` variable and change it to `$mtoolNLS.errorMandatoryEnd$`.

   In order to disable the call to the old task, comment out the following line by adding -- to the beginning of the line:

   ```
   -- <INPUT TYPE=hidden NAME="PROCESS" VALUE="CTnchs.mtool.StoreInfo">
   ```

   Now add a similar line beneath the commented out line to call the new Register2 task. This tells the form which task to use for processing.

```
<INPUT TYPE=hidden NAME="PROCESS" VALUE="CTnchs.mtool.Register2">
```

Save this file and exit. Change the permissions back to their original state.

```
> chmod 666 Register2.tem
```

4. Add the new store creation step to the mtoolTasks.xml file so that it will be recognized by Net.Commerce Hosting Server.

   The mtoolTasks.xml file contains a listing of the xml tasks that Net.Commerce Hosting Server recognizes. This file contains the necessary information about each task such as file locations, required parameters and access controls.

   To edit the mtoolTasks.xml file in the /usr/lpp/NetCommerce3/Tools/config/nchs/ directory, add write permission to the file.

   ```
   > chmod +w mtoolTasks.xml
   ```

   Open the mtoolTasks.xml file and add the following lines at the bottom of the file, just above the `</taskConfig>` line. This will register the new step in the store creation process with Net.Commerce Hosting Server and direct it to the proper files.

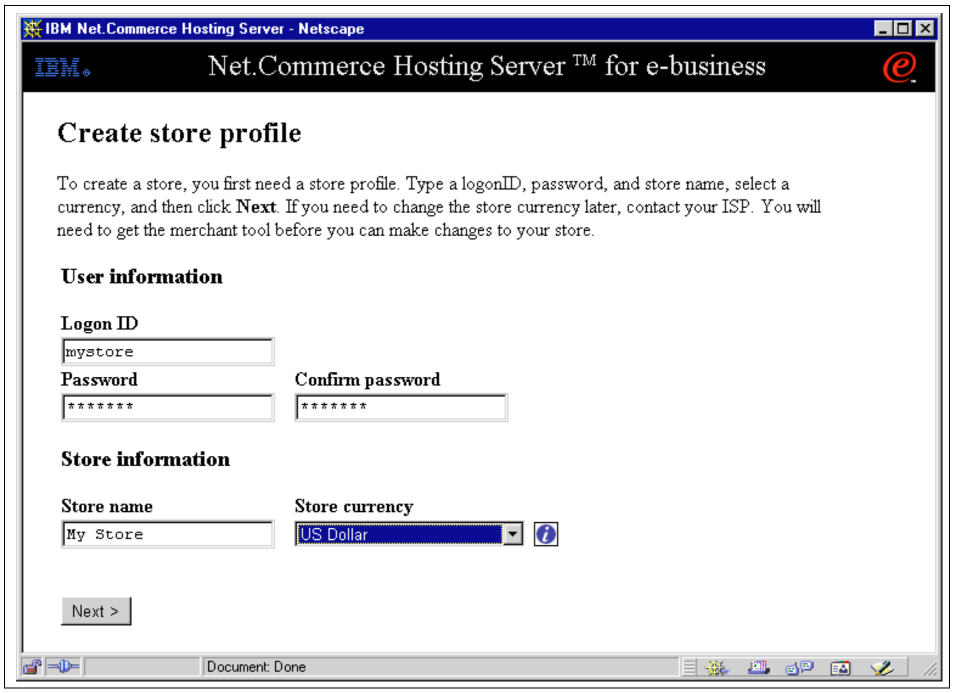
   ```
   <task name="CTnchs.mtool.Register2"
   template="nchs/mtool/Register2.tem"
   dbSessionRequired="true"
   requiredProcessParams="contactEMail1"/>
   ```

   If the CSP wants to have other mandatory fields in addition to the e-mail field, they can be added to the `requiredProcessParams` list. Save this file and exit. Change the permissions back to their original state.

   ```
   > chmod 666 mtoolTasks.xml
   ```

5. Stop and restart the Net.Commerce instance, administrator server and webserver as directed in *"Installing and Getting Started Guide",* GC09-2808-01. Figure 37 on page 88 and Figure 38 on page 89 show the

new store creation flow.


IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

Create store profile

To create a store, you first need a store profile. Type a logonID, password, and store name, select a currency, and then click **Next**. If you need to change the store currency later, contact your ISP. You will need to get the merchant tool before you can make changes to your store.

User information

Logon ID: mystore

Password: *******

Confirm password: *******

Store information

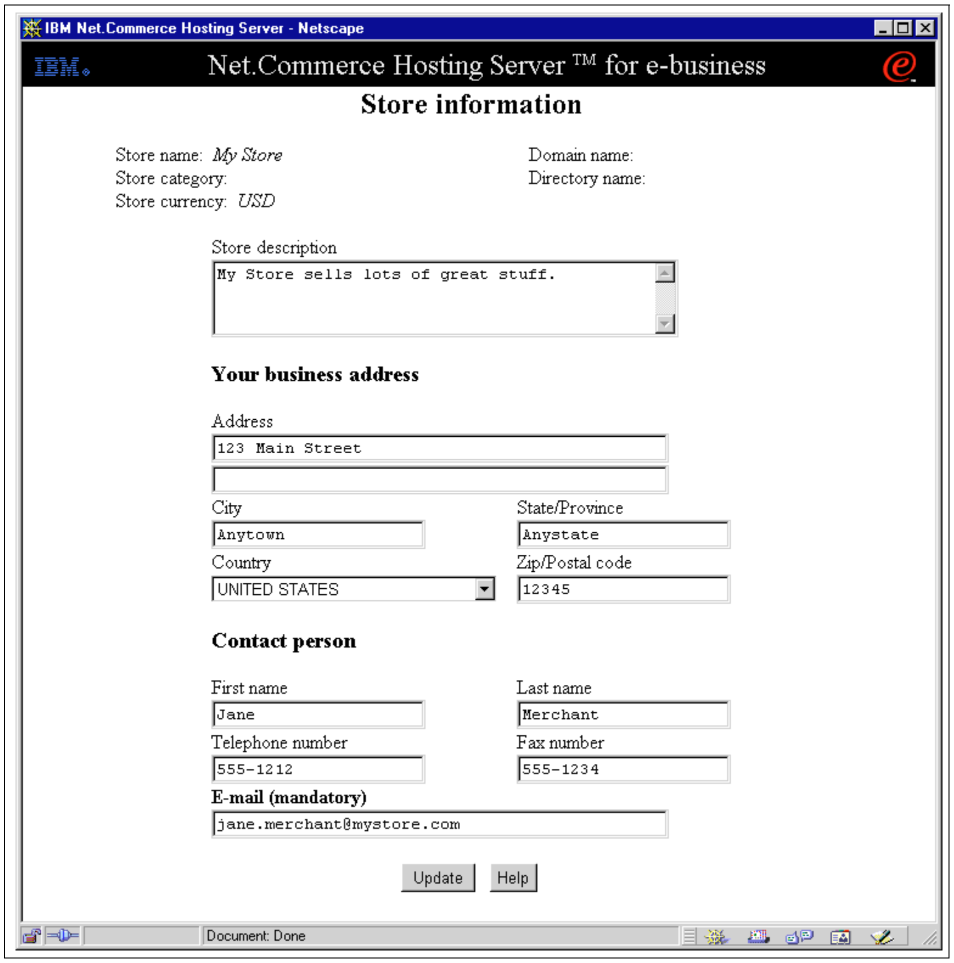
Store name: My Store

Store currency: US Dollar

Next >

Document: Done

Figure 37. Store creation form.


IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

Store information

Store name: *My Store*
Store category:
Store currency: *USD*

Domain name:
Directory name:

Store description

```
My Store sells lots of great stuff.
```

Your business address

Address	123 Main Street
City	Anytown
State/Province	Anystate
Country	UNITED STATES
Zip/Postal code	12345

Contact person

First name	Jane
Last name	Merchant
Telephone number	555-1212
Fax number	555-1234
E-mail (mandatory)	jane.merchant@mystore.com

Figure 38. Newly added Store Information form.

3.10 Restricting creation of merchant store

3.10.1 Introduction

A unique feature of Net.Commerce Hosting Server is the ability of a prospective merchant to "test drive" the software before actually purchasing the store. This gives the Commerce Service Provide, (CSP), a way to market their product more effectively. However, a CSP may want to require payment before a store can be created. This allows the CSP more control over the number of stores existing on their server and eliminates a large number of stores created just out of curiosity.

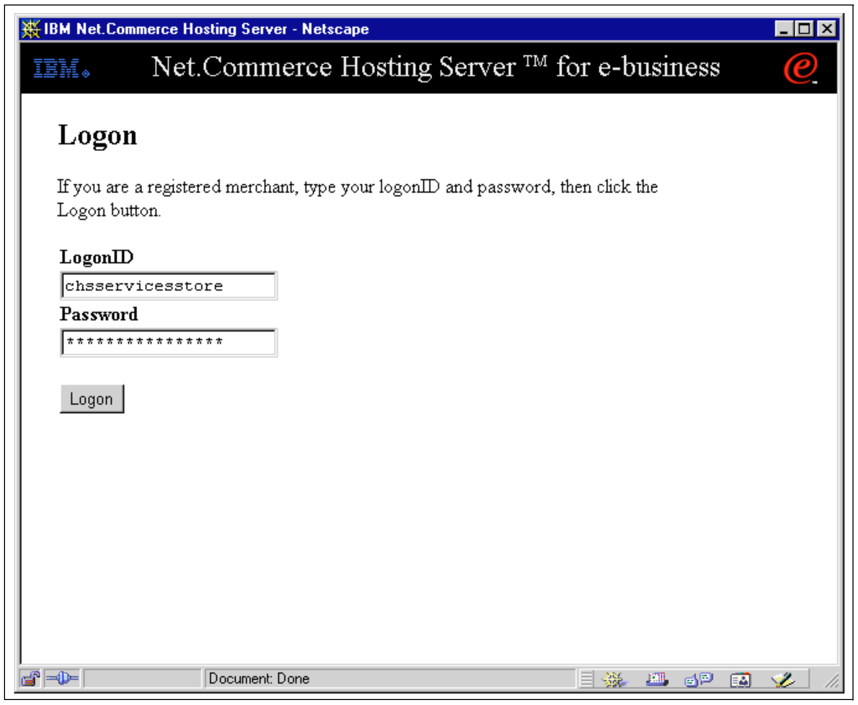
A CSP can disable this feature by setting a small fee to purchase store creation access, thus allowing the prospective merchant to "test drive" the software before actually buying the store. The **Create Store** button on the navigation bar will link to an HTML page with instructions on how to purchase store creation access. After an order has been successfully processed for store creation access, a link to the store creation function will appear on the bottom of the order confirmation page for the merchant to then create their store.

The following sections will guide you in implementing this solution on AIX. A similar guide for Windows NT can be found on Appendix G.2, “Restricting creation of merchant store on Windows NT” on page 185.

3.10.2 Restricting creation of merchant store

1. Make store creation access available for purchase from your Services Store site by adding a store creation access product to your catalog.

   Open the cspsite and click on **Manage Store**. Log on as the Commerce Hosting Server, (CHS), services store manager, (default logon and password is chsservicesstore), as shown in Figure 39.


IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

Logon

If you are a registered merchant, type your logonID and password, then click the Logon button.

LogonID: chsservicesstore

Password: ****************

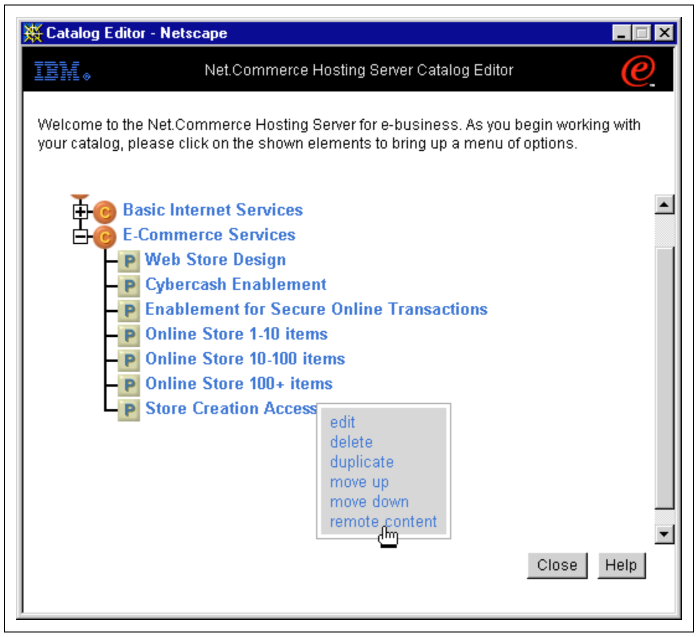
Logon

Document: Done

Figure 39. CHS services store logon screen.

Add an item to the catalog for store creation access by clicking **Edit Catalog**. Name the new item "Store Creation Access". Before exiting the **Catalog Editor**, be sure to write down the url **Add to Shopping Cart link**. It will be needed in the next step. To obtain this url, click once on the newly created Store Creation Access product and click **Remote Content**, as shown in

Figure 40. Copy the url under **Add to Shopping Cart link**.



Welcome to the Net.Commerce Hosting Server for e-business. As you begin working with your catalog, please click on the shown elements to bring up a menu of options.

*Figure 40. Obtaining the **Add to Shopping Cart link.***

In addition to needing this url for the next step, there are two pieces of information that you will need in later steps that should be obtained at this time. In the url, there are name/value pairs. Extract the store reference number and the product SKU number from these name/value pairs. For example:

```
http://<hostname>/servlet/ShoppingCart?merchant.refno=XXX&product.SKU=YYY
```

In this example, **XXX**, and **YYY** are the store reference number and product SKU number, respectively.

Exit the **Catalog Editor** and then publish the CHS Services Store by clicking **Publish Store**.

2. Change the logic of the store creation process so that a check for access privilege is made prior to allowing access to store creation.

   When the prospective merchant clicks **Create Store** the objective is to only allow access to store creation if it has been purchased. The **Create Store** link calls a static html file will displays the **Create Store** form. To disable store creation access until access has been purchased, this link will be changed to call macro instead. This macro will check to see if store creation access has been purchased and will then display the appropriate screen.

To create a macro that checks for store creation privileges, open a text editor and type the following, (adding the **Add to Shopping Cart link** where instructed):

```
%{====================================================================
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 fictitious and
any similarity to the names and addresses used by actual persons or
business enterprises is entirely coincidental.

Licensed Materials - Property of IBM
5697-D24
(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
====================================================================%}

%function(dtw_odbc) check_status() {
SELECT shfield2, shlogid
FROMshopper
WHERE shlogid='$(SESSION_ID)'

%REPORT {
%ROW {
@DTW_assign(STATUSFIELD, V_shfield2)
%}
%}
%}

%{===================================================%}
```

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

%HTML_REPORT{
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
<meta NAME="keywords"
CONTENT="Net.Commerce, Commerce Hosting Server, internet, internet
service provider, web hosting, dial up access">
<title>IBM Net.Commerce Hosting Server</title>
</head>
<body>
@check_status()
%if (STATUSFIELD != "1")
<TABLE>
<TR>
<TD>To create a store, click the link below and purchase access for only
$25!</TD>
</TR>
<TR>
<TD>Please click <A href="*** Enter Add to Shopping Cart url here
***">here</A> to purchase store creation access.</TD>
</TR>
</TABLE>
%else
<script Language="JavaScript">
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.";

function create_store() {

if ((navigator.appName.indexOf("Netscape")  > -1 &&
parseFloat(navigator.appVersion) >= 4.06) ||
(navigator.appName.indexOf("Microsoft") > -1 &&
parseFloat(navigator.appVersion) >= 4.0)) {
document.write("<p>" + launch_str+ "<br>" );
var url = "/servlet/MerchantAdmin?";
var target = window.open("", "MerchantTool",
"resizable=yes,scrollbars=yes,status=yes,width=750,height=500,screen=0,
screenY=0,left=0,top=0");

if (target.document.URL.indexOf(url) == -1)
target.location.href = url + "GOTO=Banner&body=RegisterPage";
target.focus();
} else {
document.write("<p>" + unsupported_browser_text1);
document.write("<ul>");
document.write("<li>" + unsupported_browser_item1 + "</li>");
document.write("<li>" + unsupported_browser_item2 + "</li>");
document.write("</ul>");
document.write(unsupported_browser_text2);
}
}

create_store();
</script>
%endif

<br>
<br>
<br>
<br>
<font SIZE="1" FACE="Verdana, Arial, Helvetica">Please send all
inquiries to:&nbsp;<a
HREF="mailto:webmaster@CHSNet.com">webmaster@CHSNet.com</a><br>
Site comments:&nbsp;<a
HREF="mailto:webmaster@CHSNet.com">webmaster@CHSNet.com</a><br>
© 1998, IBM Net.Commerce for CHS.<br>
</font></p>
</body>
</html>
%}
```

Save this file as creation_access.d2w in the /usr/lpp/NetCommerce3/macro/en_US/ncadmin/sitemgr/ directory. Make sure the file has the correct permissions set by issuing the following commands:

```
> cd /usr/lpp/NetCommerce3/macro/en_US/ncadmin/sitemgr/
> chmod 644 ./creation_access.d2w
```

3. Edit the navigation.html file to change the link for **Create Store**.

   The **Create Store** link in the navigation.html file calls a static html file will displays the **Create Store** form. To disable store creation access until access has been purchased, this link will be changed to call the newly created creation_access.d2w macro instead.

   Use a text editor to open the file navigation.html in the /usr/lpp/NetCommerce3/html/en_US/cspsite/ directory.

   In order to change the **Create Store** link, comment out the link to the create_store.html file by placing HTML comment tags around this section:

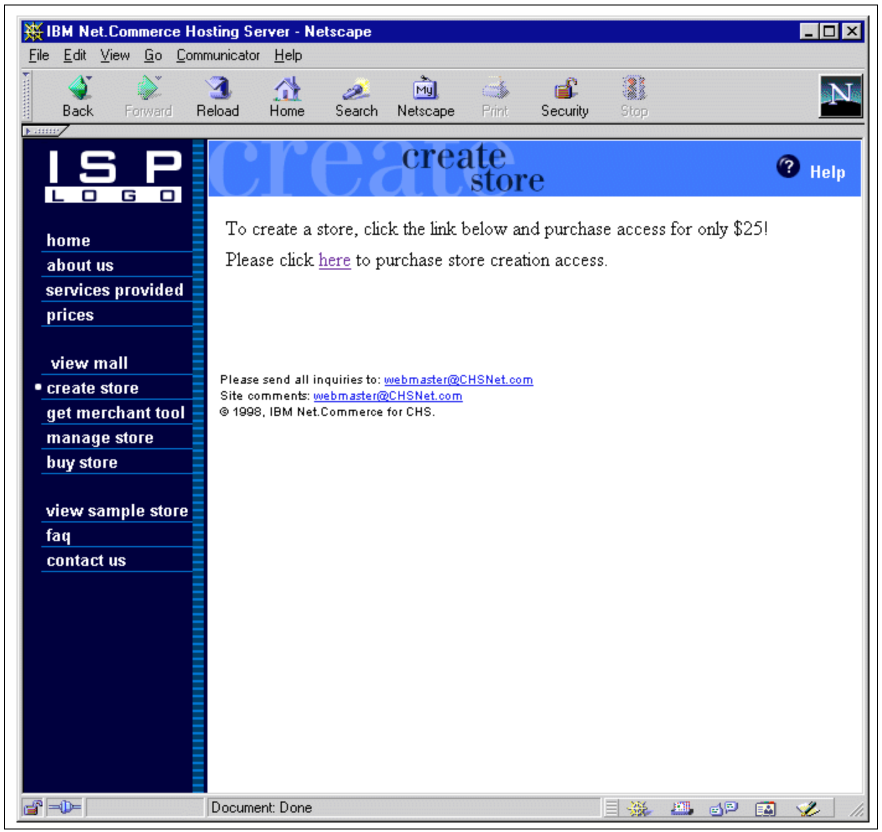
```
<!-- create store -->
<!-- <TR>
<TD><IMG NAME="item6" SRC="/CHS/images/bullet_blank.gif" WIDTH=5
HEIGHT=5 ALT="" BORDER=0></TD>
<TD NOWRAP>&nbsp;<B><A
HREF="javascript:go(6,'/cspsite/create_store_banner.html','/cspsite/cre
ate_store.html')" onMouseOver="on(6); status='create store'; return
true;" onMouseOut="off(6); status='';">create store</A></B></TD>
</TR>
<TR>
<TD></TD>
<TD><IMG SRC="/CHS/images/separator.gif" WIDTH=122 HEIGHT=1 ALT=""
BORDER=0></TD>
</TR> -->
```

   Replace this commented out section with a new, similar section containing a link to the creation_access.d2w macro:

```
<TR>
<TD><IMG NAME="item6" SRC="/CHS/images/bullet_blank.gif" WIDTH=5
HEIGHT=5 ALT="" BORDER=0></TD>
<TD NOWRAP>&nbsp;<B><A HREF="javascript:go(6,
'/cspsite/create_store_banner.html',
'/cgi-bin/ncommerce3/ExecMacro/ncadmin/sitemgr/creation_access.d2w/repo
rt?merfnb=$env.merchant_id$')" onMouseOver="on(6); status='create
store'; return true;" onMouseOut="off(6); status='';">create
store</A></B></TD>
```

```
</TR>
<TR>
<TD></TD>
<TD><IMG SRC="/CHS/images/separator.gif" WIDTH=122 HEIGHT=1 ALT=""
BORDER=0></TD>
</TR>
```

Save the navigation.html file and exit. Figure 41 shows the new screen that will appear when a merchant clicks **Create Store**.



To create a store, click the link below and purchase access for only $25!

Please click here to purchase store creation access.

Please send all inquiries to: webmaster@CHSNet.com
Site comments: webmaster@CHSNet.com
© 1998, IBM Net.Commerce for CHS.

Figure 41. New store creation access screen.

4. Modify the ord_ok.d2w macro to enable store creation after access has been purchased.

   The ord_ok.d2w macro displays the order confirmation screen. When store creation access has been purchased, the shfield2 field in the shopper table should be set to 1 to indicate that the prospective merchant has purchased access and can therefore be granted access to store creation.

This update can be performed by checking the list of products purchased in the product list and if any of those products is the store creation access product, the shfield2 field is set to 1 indicating access has been purchased. After this update to the database has been performed, a link to the store creation command will be displayed so that the merchant can then create their store. This link is only available when store creation access has been purchased and is only active for that particular session. This prevents a registered merchant from returning to the cspsite and creating additional stores.

To add this functionality to the ord_ok.d2w macro, open this file which is located in the /usr/lpp/NetCommerce3/macro/en_US/`<ref num>`/ directory, (where `<ref num>` is the reference number of the CHS Services Store obtained in Step 1).

In this file, make the following changes which appear in **boldface**, (replace all `<ref num>` occurrences with the CHS Services Store reference number).

```
%include "<ref num>/include.inc"
%include "<ref num>/new_ord_ok.d2w"
```

Save this file and exit. The changes made to the ord_ok.d2w macro instruct Net.Commerce to include a new macro called new_ord_ok.d2w. This macro will be very similar to the existing ord_ok.d2w macro in the /usr/lpp/NetCommerce3/macro/common/CSPstoremodel/ directory except for the addition of the logic to update the database with a flag to grant store creation access.

In order to create this new macro, make a copy of the ord_ok.d2w macro in the /usr/lpp/NetCommerce3/macro/common/CSPstoremodel/ directory and name this copy new_ord_ok.d2w. Move this new copy to the /usr/lpp/NetCommerce3/macro/en_US/`<ref num>`/ directory, (where `<ref num>` is the reference number of the CHS Services Store). Open the new_ord_ok.d2w macro file in a text editor and make the following additions/changes which appear in **boldface**. Insert the product SKU number obtained in Step 1 where instructed.

```
%{===============================================================
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 fictitious and
any similarity to the names and addresses used by actual persons or
business enterprises is entirely coincidental.

Licensed Materials - Property of IBM 5697-D24 (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
==================================================================%}
%INCLUDE "/CSPstoremodel/translation_text.inc"
%INCLUDE "/CSPstoremodel/format_pricedefinition.inc"
%INCLUDE "$(DirectoryName)/navbar.inc"

%define {
SHOWSQL="NO"
CreationFlag="False"
%}

%INCLUDE "ord_taxshiprules.inc"

%function(dtw_odbc) UPDATE_CREATION_ACCESS(){

UPDATEshopper
SET shfield2 = '1'
WHERE shlogid = '$(SESSION_ID)'
%}

%function(dtw_odbc) GET_ORBILLTO (){
SELECT orbllto
FROM orders
WHERE orrfnbr = $(order_rn)

%REPORT {
%ROW {
@dtw_assign(BILLING_ADDRESS_RN, V_orbllto)
%}
%}
```

```
%MESSAGE{
100: { %} :CONTINUE
default: { ERROR in GET_ORBLLTO %}
%}
%}

%function(dtw_odbc) IS_SET ()
{
SELECT ompaymthd, setsstatcode, setsfailtype
FROM ordpaymthd, setstatus, orders
WHERE omornbr = $(order_rn) and setsornbr = $(order_rn) and orrfnbr =
setsornbr and orshnbr = $(SESSION_RN)

%REPORT {
<CENTER>
<TABLE width=530 CELLPADDING=4 CELLSPACING=0 BORDER=0 ALIGN="center">
<TR>
<TD ALIGN="left" VALIGN="center">

%ROW {
@dtw_assign(BILLING_ADDRESS_RN, V_orbllto)
@dtw_assign(PAYMENT_METHOD, V_ompaymthd )
<B>$(TXT_THANKYOU)</B>
<BR>
%INCLUDE "ord_set_returncodes.inc"
%}

</TD>
</TR>
</TABLE>
%}
%MESSAGE{
100: { %} :CONTINUE
default: { ERROR in IS_SET %}
%}
%}

%function(dtw_odbc) GET_SHOPPER_TYPE() {
select shshtyp
from shopper
where shrfnbr = $(SESSION_RN)

%REPORT{
```

```
%ROW{
 @DTW_assign(SHOPPER_TYPE, V_shshtyp)
%}
%}
%MESSAGE{
default: { ERROR in GET_SHOPPER_TYPE %}
%}
%}

%function(dtw_odbc) SHOPPER_INFO() {
select sarfnbr, salname, safname, saaddr1, saaddr2, sacity, sastate,
sazipc, sacntry
from shaddr
where sashnbr=$(SESSION_RN) and sarfnbr=$(BILLING_ADDRESS_RN)

%REPORT{
<CENTER>
<TABLE width=530 CELLSPACING=0 CELLPADDING=4 BORDER=0 ALIGN="center">

%ROW{
%IF (($(PAYMENT_METHOD) != "SET") && ($(PAYMENT_METHOD) != "SETNV")
<TR>
<TD COLSPAN=3>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
<B>$(TXT_THANKYOU)</B>
   </FONT>
   </TD>
   </TR>
   <TR>
<TD COLSPAN=3>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CONF_MSG)</FONT>
</TD>
</TR>
%ENDIF

<TR><TD><BR><BR></TD></TR>
<TR>
    <TD COLSPAN=3 ALIGN="center" bgcolor="#E0E0E0">
<B>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
$(LBL_ORDERNUMBER) : $(order_rn)
%IF (SHOPPER_TYPE == "G")Z--
$(LBL_CUSTOMERCODE) :  $(SESSION_ID)
%ENDIF
</FONT>
</B>
</TD>
</TR>
<TR><TD><BR></TD></TR>
   <TR>
<TD>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(TXT_MAILTO)
 </B></FONT>
</TD>
<TD width=10></TD>
<TD>
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(TXT_MAILSHIPTO)</B></FO
 NT>
</TD>
</TR>
   <TR>
<TD BGCOLOR=white width=260 VALIGN=top>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
%INCLUDE "/CSPstoremodel/address_static.inc"
</FONT>
</TD>
<TD width=10></TD>
 %}
 %}
 %MESSAGE{default: { ERROR in SHOPPER_INFO %} :CONTINUE
 %}
 %}

 %function(dtw_odbc) SHOPPER_SHIPTO_INFO() {
 select salname, safname, saaddr1, saaddr2, sacity, sastate, sazipc,
 sacntry
 from shaddr, shipto
 where stornbr=$(order_rn) and stsanbr=sarfnbr

 %REPORT{

<TD BGCOLOR=white  width=260 VALIGN=top>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
%INCLUDE "/CSPstoremodel/address_static.inc"
</FONT>
```

```
</TD>
</TR>
 </TABLE>
 </CENTER>
 %}
 %MESSAGE{default: { ERROR in SHOPPER_SHIPTO_INFO %}
 %}
 %}

 %function(dtw_odbc) DISPLAY_DETAILS_LIST() {
 select strfnbr, stsanbr, stshnbr, stmenbr, stprnbr, stprice, stquant,
 stcpcur,
prrfnbr, prnbr, prldesc2, prsdesc, salname, safname
 from shipto, product, shaddr
 wherestshnbr=$(SESSION_RN) and stmenbr=$(MerchantRefNum) and
 stprnbr=prrfnbr and stornbr=$(order_rn)
and stsanbr=sarfnbr
 order by stmenbr, stsanbr, strfnbr

 %REPORT{
 <BR>
 <CENTER>
 <TABLE width=530 CELLPADDING=4 CELLSPACING=0 BORDER=0 ALIGN="center">
 <TR>
     <TD ALIGN=left VALIGN=top><B><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_PRODUCTNUM)</FONT></B>
 </TD>
<TD ALIGN=left VALIGN=top><B><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_PRODUCTNAME)</FONT></B
 ></TD>
<TD ALIGN=middle VALIGN=top><B><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_QUANTITY)</FONT></B></
 TD>
     <TD ALIGN=right VALIGN=top><B><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_PRODUCTPRICE)
%IF (CurDescription != null)
<BR>[$(CurDescription)]
%ENDIF
</FONT></B></TD>
 <TD ALIGN=right VALIGN=top><B><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_SUBTOTAL)
%IF (CurDescription != null)
<BR>[$(CurDescription)]
```

```
%ENDIF
</FONT></B></TD>
</TR>
<TR><TD colspan=5><HR></TD></TR

 %ROW{
<TR>
@DTW_FORMAT(V_stprice, "", CurDecimalPlaces, FORMATTEDPRODPRICE)
@DTW_MULTIPLY(V_stquant, V_stprice, SUB_TOT)
@DTW_FORMAT(SUB_TOT, "", CurDecimalPlaces, FORMATTEDSUBTOTPRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDSUBTOTPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSUBTOTPRICE,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDPRODPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDPRODPRICE,OUT_PRICE)

%if ($(V_prnbr) == "**insert product SKU number here**")
@DTW_ASSIGN(CreationFlag, "True")
%endif

    <TD ALIGN=left><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(V_prldesc2) - </FONT></TD>
<TD ALIGN=left><FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
 $(V_prsdesc)</FONT></TD>
<TD ALIGN=middle><FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
 $(V_stquant)</FONT></TD>
    <TD ALIGN=right><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDPRODP
 RICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_stcpcur)
%ENDIF
</FONT></TD>
    <TD ALIGN=right ALIGN="right"><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(CurPrefix)$(FORMATTEDSU
 BTOTPRICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_stcpcur)
%ENDIF
</B></FONT></TD>
</TR>
<TR><TD HEIGHT=5></TD></TR>
```

```
%}

<TR><TD colspan=5><HR></TD></TR>
%}
%MESSAGE{
100    : {<BR><FONT
SIZE=3><B>$(MSG_ORDERLIST_EMPTY)</B></FONT>%}:continue
default: {ERROR : Problem with DISPLAY_DETAILS_LIST function %}
%}
%}

%function(dtw_odbc) DISPLAY_CHARGES_MerchantTax() {
select distinct orprtot, ortxtot, orshtot, orshtxtot, orcpcur,
(oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as mttax
from orders, orderpay, shaddr
whereormenbr=$(MerchantRefNum) and oysanbr=sarfnbr and
orrfnbr=$(order_rn) and oyornbr=$(order_rn)
%REPORT{

%ROW{
@DTW_FORMAT(V_orprtot, "", CurDecimalPlaces, FORMATTEDSUBTOTPRICE)
@DTW_FORMAT(V_ortxtot, "", CurDecimalPlaces, FORMATTEDTAXTOT)
@DTW_FORMAT(V_orshtot, "", CurDecimalPlaces, FORMATTEDSHIPTOT)
@DTW_FORMAT(V_orshtxtot, "", CurDecimalPlaces, FORMATTEDSHIPTAXTOT)
@DTW_ADD(V_orprtot, V_ortxtot, total)
@DTW_ADD(total, V_orshtot, total)
@DTW_ADD(total, V_orshtxtot, total)
@DTW_FORMAT(total, "", CurDecimalPlaces, FORMATTEDTOTPRICE)
%IF (ConvMultOrDiv == "M")
@DTW_MULTIPLY(FORMATTEDTOTPRICE, ConvFactor, CONVPRICE)
@DTW_FORMAT(CONVPRICE, "", ConvCurDecimalPlaces, CONVFORMATTEDPRICE)
%ELIF (ConvMultOrDiv == "D")
@DTW_DIVIDE(FORMATTEDTOTPRICE, ConvFactor, CONVPRICE)
@DTW_FORMAT(CONVPRICE, "", ConvCurDecimalPlaces, CONVFORMATTEDPRICE)
%ENDIF
@DTW_ASSIGN(IN_PRICE,FORMATTEDSUBTOTPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSUBTOTPRICE,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDTAXTOT)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDTAXTOT,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDSHIPTOT)
%INCLUDE "/CSPstoremodel/format_price.inc"
```

```
@DTW_ASSIGN(FORMATTEDSHIPTOT,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDSHIPTAXTOT)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSHIPTAXTOT,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDTOTPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDTOTPRICE,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,CONVFORMATTEDPRICE)
%INCLUDE "/CSPstoremodel/format_convprice.inc"
@DTW_ASSIGN(CONVFORMATTEDPRICE,OUT_PRICE)
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_SUBTOTAL)
 </B></FONT></TD>
    <TD ALIGN="right"><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSUBTO
 TPRICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_orcpcur)
%ENDIF
</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_TAX)
 </B></FONT></TD>
    <TD ALIGN="right">
%IF (TAXRULE_EXISTS == "YES" && CurDescription == null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDTAXTO
 T)$(CurPostfix) $(V_orcpcur)</FONT>
%ELIF (TAXRULE_EXISTS == "YES" && CurDescription != null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDTAXTO
 T)$(CurPostfix)</FONT>
%ELSE
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>-----</FONT>
%ENDIF
</TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_SHIPPING)
 </b></FONT></TD>
     <TD ALIGN="right">
%IF (SHIPRULE_EXISTS == "YES" && CurDescription == null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 OT)$(CurPostfix) $(V_orcpcur)</FONT>
%ELIF (SHIPRULE_EXISTS == "YES" && CurDescription != null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 OT)$(CurPostfix) </FONT>
%ELSE
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>-----</FONT>
%ENDIF
</TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_SHIPPINGTAX)
 </b></FONT></TD>
     <TD ALIGN="right">
%IF (SHIPRULE_EXISTS == "YES" && CurDescription == null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 AXTOT)$(CurPostfix) $(V_orcpcur)</FONT>
%ELIF (SHIPRULE_EXISTS == "YES" && CurDescription != null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 AXTOT)$(CurPostfix)</FONT>
%ELSE
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>-----</FONT>
%ENDIF
</TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_TOTAL)
 </B></FONT></TD>
     <TD ALIGN="right" BGCOLOR="white"><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(CurPrefix)$(FORMATTEDTO
 TPRICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_orcpcur)
%ENDIF
```

```
</B></FONT></TD>
</TR>
<TR><TD HEIGHT=5></TD></TR>
<TR><TD><BR></TD></TR>
<TR BGCOLOR="#E0E0E0">
<TD COLSPAN=5 ALIGN=center>
<FONT size=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
<B>$(TXT_YOUCHARGED)</B>
%IF (ConvMultOrDiv == "")
<B>$(CurPrefix)$(FORMATTEDTOTPRICE)$(CurPostfix) $(V_orcpcur)</B>
%ELSE
<B>$(CurPrefix)$(FORMATTEDTOTPRICE)$(CurPostfix) $(CurDescription)</B>
 [$(ConvCurPrefix)$(CONVFORMATTEDPRICE)$(ConvCurPostfix)
 $(ConvCurDescription)]
%ENDIF
</FONT>
</TD>
</TR>
%}
  </TABLE>
  </CENTER>
  <BR>
 %}
 %MESSAGE{
 100: {  No  Information Available. %} : continue
     default: { ERROR in DISPLAY_CHARGES_MerchantTax() %}:CONTINUE
 %}
 %}

 %function(dtw_odbc) GET_CONF_MESSAGE() {
 SELECT omornbr,ompaymthd,ompaydevc,pmentinst2,pmentinst4
 FROM   ordpaymthd,merpayinfo
 WHERE  paymerid=$(MerchantRefNum) and omornbr=$(order_rn)

 %REPORT {
 %ROW {
 %if (V_ompaydevc == "OFF-LINE")
 @DTW_assign(CONF_MSG, V_pmentinst2)
 %else
 @DTW_assign(CONF_MSG, V_pmentinst4)
 %endif
 %}
 %}
```

```
%MESSAGE{
default: {<FONT COLOR="red">error occurred in
GET_CONF_MESSAGE()</font>%}:CONTINUE
%}
%}

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

%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT"></HEAD>
<TITLE>$(LongStoreName) [$(TXT_TITLE_ORDEROK)]</TITLE>
<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)"
TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
%IF (CurDecimalPlaces == "")
@DTW_assign(CurDecimalPlaces, "2")
%ENDIF
%INCLUDE "/CSPstoremodel/assign_priceseparators.inc"
<CENTER>
<TABLE width=530 CELLPADDING=4 CELLSPACING=0 BORDER=0 ALIGN="center">
<TR>
<TD ALIGN="center" VALIGN="center">
<FONT
COLOR=$(TextCol)><H$(DONOTTRANSLATE_FORMAT_FONTSIZETITLE)>$(TXT_TITLE_O
RDEROK)</H$(DONOTTRANSLATE_FORMAT_FONTSIZETITLE)></FONT>
</TD>
</TR>
</TABLE>
</CENTER>
@IS_SET()
%IF (($(PAYMENT_METHOD) == "SET") ||($(PAYMENT_METHOD) == "SETNV") )
@SET_TAXRULE_FLAG()
@SET_SHIPRULE_FLAG()
%ELSE
@GET_CONF_MESSAGE()
%ENDIF
@GET_SHOPPER_TYPE()
@GET_ORBILLTO()
@SHOPPER_INFO()
@SHOPPER_SHIPTO_INFO()
@DISPLAY_DETAILS_LIST()
@DISPLAY_CHARGES_MerchantTax()

%if (CreationFlag == "True")
@UPDATE_CREATION_ACCESS()
<center><a
href="/cgi-bin/ncommerce3/ExecMacro/ncadmin/sitemgr/creation_access.d2w
/report?merfnb=$env.merchant_id$">Click here to create your on-line
store!</a></center>
%endif

@DISPLAY_CUSTOM_NAVBAR()
</body>
</html>
%}
```

Save this file and exit. Make sure the file permissions are set correctly by issuing the following commands, (replace `<ref num>` with the store reference number):

```
> cd /usr/lpp/NetCommerce3/macro/en_US/<ref num>/
> chmod 644 new_ord_ok.d2w
```

This new macro will update the database with a flag granting store creation access if it has been purchased.

5. Stop and restart the Net.Commerce instance, administrator server and webserver as directed in *"Installing and Getting Started Guide",* GC09-2808-01.

Appendix A. Net.Data macro for the category items page

This appendix contains customized version of the csp_cat.d2w macro file found in /usr/lpp/NetCommerce3/macro/en_US/category/CSPstoremodel directory. It is used to customize default shopping flow.

csp_cat.d2w

```
%{========================================================================

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

5697-D24

(c)  Copyright  IBM Corp.  1998.      All Rights Reserved

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

========================================================================%}

%INCLUDE "/CSPstoremodel/translation_text.inc"

%define {
  SHOWSQL="NO"
ATTRIBUTES = "FALSE"
BACKUP = "$(HomeCategory)"
%}

%function(dtw_odbc) GET_ADDRESS_REF_NUM() {
   select sarfnbr
   from shaddr, shopper
   where (shlogid='$(SESSION_ID)' and sanick=shlogid and shrfnbr=sashnbr and
saadrflg='P')
   %REPORT{
     %ROW{
   @DTW_assign(ADDRESS_REF, V_sarfnbr)
     %}
   %}
   %MESSAGE{
     default: {%}: continue
   %}
%}

%function(dtw_odbc) GET_CATEGORY_BANNERINFO(){
    select distinct cgname, cgrfnbr
    from CATEGORY
    where cgrfnbr=$(cgrfnbr) and cgmenbr=$(cgmenbr)
```

```
%REPORT{

    %ROW{
@DTW_assign(CGRYNUM, V_CGRFNBR)
@DTW_assign(CGRYBANNERNAME, V_cgname)
    %}


  %}
  %MESSAGE{100:{erro %} :continue %}
%}

%{==== DISPLAY_CATEGORIES Function ====%}

%function(dtw_odbc) DISPLAY_CATEGORIES(){
    select CATEGORY.CGRFNBR, CATEGORY.CGMENBR, CATEGORY.CGNAME, CATEGORY.CGTHMB,
CGRYREL.CRSEQNBR, CATEGORY.CGLDESC, CGRYREL.CRPCGNBR
    from CATEGORY, CGRYREL
    where CRCCGNBR=CGRFNBR and crpcgnbr=$(cgrfnbr) and crmenbr=$(cgmenbr) and cgpub=1
    order by crseqnbr

%REPORT{

<CENTER>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=2 WIDTH=530>
  <TR><TD><H3>$(TXT_CATALOGINDEX)</H3></TD></TR>


%ROW{

<TR>
<TD ALIGN="left" VALIGN="top" WIDTH=75>
<ul>
<li>
<A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_CGRFNBR)&cgmenbr=$(V_CGMENBR)&CGRY
_NUM=$(CGRYNUM)">
<B>$(V_cgname)</B></A>
</li>
</ul>
</TD>
</TR>

<TR><TD HEIGHT=5></TD></TR>

    %}

</TABLE>
<CENTER>

  %}
  %MESSAGE{100:{%} :continue %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_LIST() {

    select PRODUCT.PRRFNBR, PRODUCT.PRMENBR, PRODUCT.PRFULL, PRODUCT.PRNBR, PRSDESC,
PRTHMB, PRLDESC1,
CGPRREL.CPSEQNBR, CGPRREL.CPCGNBR,
FULLWD, FULLHT, FULLAL, FULLTXT,
                PPPRC, PANAME, PAVAL
    from PRODUCT, CGPRREL, PRODIMAGE, PRODPRCS, PRODATR
    where CPPRNBR=PRRFNBR and cpcgnbr=$(cgrfnbr) and cpmenbr=$(cgmenbr) and prpub=1
and merchant_rn=$(MerchantRefNum) and product_rn=prrfnbr
                        and ppmenbr=merchant_rn and cpmenbr=ppmenbr
                        and ppprnbr=prrfnbr
                        and prmenbr=ppmenbr and prmenbr=pamenbr and paprnbr=prrfnbr
    order by cpseqnbr, prrfnbr, cpseqnbr

%REPORT{

<CENTER>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=530>

<TR><TD><H3>$(TXT_CATEGORYITEMS)</H3></TD></TR>

@DTW_assign(pre_rrfnbr, "null")
@DTW_assign(firstTime, "true")


%ROW{

@DTW_assign(firstRrfnbr, V_PRRFNBR)

<TR>
<TD ALIGN="left" VALIGN="top" WIDTH=180>

%IF ( pre_rrfnbr != "null" && V_prrfnbr != prr_no )

<!--// delete "Order" button //-->
<!--<FORM ACTION="/cgi-bin/ncommerce3/ProductDisplay" METHOD="post">
<INPUT TYPE=hidden NAME="prmenbr" VALUE="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="prrfnbr" VALUE="$(pre_rrfnbr)">
<INPUT TYPE=hidden NAME=product_rn VALUE=$(pre_rrfnbr)>
<INPUT TYPE=SUBMIT VALUE="$(BUT_ORDER)">
</FORM> -->

<!--// inserted for "Quick order" button  //-->
<br><FORM NAME="process" ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" Method=get>
<INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>


                         <INPUT TYPE=hidden   NAME=product_rn   VALUE=$(pre_rrfnbr)>
                         <INPUT TYPE=hidden   NAME=quantity     VALUE=1>
                         <INPUT TYPE=hidden   NAME=url
VALUE=/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchantRefNum)>

                        <TEXTAREA NAME="comment" ROWS="4" COLS="60"></TEXTAREA>
                        <br><br>
<INPUT TYPE=SUBMIT VALUE="Quick Order">
                </FORM>
<!--// end of insertion //-->

<CENTER>$(HorizontalSep)</CENTER>
%ENDIF

%IF ( V_prrfnbr != prr_no )
<BR><BR>
<!--A
HREF="/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=$(V_PRRFNBR)&prmenbr=$(V_PRMENBR)&CGRY_
NUM=$(CGRYNUM)"-->
<FONT SIZE=3><B>$(V_PRSDESC)</B></FONT>
%ENDIF
</TD>
```

```
</TR>
<TR>
<TD>
%IF (V_fullal == "0" && V_prfull != "" && V_prrfnbr != prr_no  && pre_rrfnbr == "null" &&
firstTime== "true" )
@DTW_assign(firstTime, "false")
<TABLE WIDTH=530 BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR>
<TD>
<FORM ACTION="/cgi-bin/ncommerce3/ProductDisplay" METHOD="get">
<INPUT TYPE=hidden NAME="prmenbr" VALUE="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="prrfnbr" VALUE="$(firstRrfnbr)">
<INPUT TYPE=hidden NAME=product_rn VALUE=$(pre_rrfnbr)>
<INPUT TYPE=image  SRC="$(V_PRFULL)" BORDER=0 WIDTH=$(V_fullwd) HEIGHT=$(V_fullht)
ALT="$(V_fulltxt)" HSPACE=0>
</FORM>

</TD>
<TD>
<FONT SIZE=2>$(V_PRLDESC1)</FONT>
</TD>
</TR>

</TABLE>

%ELIF ( V_fullal == "0" && V_prfull != "" && V_prrfnbr != prr_no && pre_rrfnbr != "null")

<TABLE WIDTH=530 BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR>
<TD>
<FORM ACTION="/cgi-bin/ncommerce3/ProductDisplay" METHOD="get">
<INPUT TYPE=hidden NAME="prmenbr" VALUE="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="prrfnbr" VALUE="$(V_prrfnbr)">
<INPUT TYPE=hidden NAME=product_rn VALUE=$(pre_rrfnbr)>
<INPUT TYPE=image  SRC="$(V_PRFULL)" BORDER=0 WIDTH=$(V_fullwd) HEIGHT=$(V_fullht)
ALT="$(V_fulltxt)" HSPACE=0>
</FORM>
</TD>
<TD>
<FONT SIZE=2>$(V_PRLDESC1)</FONT>
</TD>
</TR>
</TABLE>

%ELIF (V_fullal == "1" && V_prfull != "" && V_prrfnbr != prr_no)

<TABLE WIDTH=530 BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR>
<TD>
<FONT SIZE=2>$(V_PRLDESC1)</FONT>
</TD>
<TD>
<IMG SRC="$(V_PRFULL)" BORDER=0 WIDTH=$(V_fullwd) HEIGHT=$(V_fullht) ALT="$(V_fulltxt)"
HSPACE=0>
</TD>

</TR>
</TABLE>

%ELIF (V_fullal == "2" && V_prfull != "" && V_prrfnbr != prr_no )

<TABLE WIDTH=530 BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR>
```

```
<TD>
<CENTER>
<IMG SRC="$(V_PRFULL)" BORDER=0 WIDTH=$(V_fullwd) HEIGHT=$(V_fullht) ALT="$(V_fulltxt)"
 HSPACE=0>
</CENTER>
</TD>
</TR>
<TR>
<TD>
<FONT SIZE=2>$(V_PRLDESC1)</FONT>
</TD>
</TR>
</TABLE>

%ELIF (V_fullal == "3" && V_prfull != "" && V_prrfnbr != prr_no )

<TABLE WIDTH=530 BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR>
<TD>
<FONT SIZE=2>$(V_PRLDESC1)</FONT>
</TD>
</TR>
<TR>
<TD>
<CENTER>
<IMG SRC="$(V_PRFULL)" BORDER=0 WIDTH=$(V_fullwd) HEIGHT=$(V_fullht) ALT="$(V_fulltxt)"
 HSPACE=0>
</CENTER>
</TD>
</TR>
</TABLE>

%ELIF ( V_prrfnbr != prr_no )

<TABLE WIDTH=530 BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR>
<TD>
<FONT SIZE=2>$(V_PRLDESC1)</FONT>
</TD>
</TR>
</TABLE>

%ELSE
%ENDIF


%IF ( V_prrfnbr != prr_no  )
@DTW_assign(prr_no, V_prrfnbr)
<BR><I>Price:</I>  $(V_PPPRC)<BR>
%ENDIF

%IF ( V_prrfnbr != prr_no && $(V_PANAME) != "dummy"  && $(V_PAVAL) != ""  )
<I>$(V_PANAME):</I> $(V_PAVAL)<BR>
%ELIF ( V_prrfnbr == prr_no && $(V_PANAME) != "dummy"  && $(V_PAVAL) != "" )
<I>$(V_PANAME):</I> $(V_PAVAL)<BR>
%ENDIF

@DTW_assign(pre_rrfnbr, V_prrfnbr)

%IF ( V_prrfnbr != prr_no )

 <!--// delete "Order" button //-->
 <!--<FORM ACTION="/cgi-bin/ncommerce3/ProductDisplay" METHOD="get">
```

```
<INPUT TYPE=hidden NAME="prmenbr" VALUE="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="prrfnbr" VALUE="$(V_prrfnbr)">
<INPUT TYPE=hidden NAME=product_rn VALUE=$(V_prrfnbr)>
<INPUT TYPE=SUBMIT VALUE="$(BUT_ORDER)">
</FORM>  -->


<!--// inserted for "Quick order" button  //-->
<br><FORM NAME="process" ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" Method=get>
                        <INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>


                         <INPUT TYPE=hidden   NAME=product_rn   VALUE=$(pre_rrfnbr)>
                         <INPUT TYPE=hidden   NAME=quantity     VALUE=1>
                         <INPUT TYPE=hidden   NAME=url
VALUE=/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchatRefNum)>

                        <TEXTAREA NAME="comment" ROWS="4" COLS="60"></TEXTAREA>
                        <br><br>
                        <INPUT TYPE=SUBMIT VALUE="Quick Order">
                </FORM>
<!--// end of insertion //-->


<CENTER>$(HorizontalSep)</CENTER>
%ENDIF

</TD>
</TR>
%}

<TR>
<TD>

<!--// delete "Order" button //-->
<!--<FORM ACTION="/cgi-bin/ncommerce3/ProductDisplay" METHOD="get">
<INPUT TYPE=hidden NAME="prmenbr" VALUE="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="prrfnbr" VALUE="$(pre_rrfnbr)">
<INPUT TYPE=hidden NAME=product_rn VALUE=$(pre_rrfnbr)>
<INPUT TYPE=SUBMIT VALUE="$(BUT_ORDER)">
</FORM> -->


<!--// inserted for "Quick order" button  //-->
<br><FORM NAME="process" ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" Method=get>
                        <INPUT TYPE=hidden   NAME=merchant_rn  VALUE=$(MerchantRefNum)>

                         <INPUT TYPE=hidden   NAME=product_rn   VALUE=$(pre_rrfnbr)>
                         <INPUT TYPE=hidden   NAME=quantity     VALUE=1>
                         <INPUT TYPE=hidden   NAME=url
VALUE=/cgi-bin/ncommerce3/OrderItemList?merchant_rn=$(MerchantRefNum)>

                        <TEXTAREA NAME="comment" ROWS="4" COLS="60"></TEXTAREA>
                        <br><br>
                        <INPUT TYPE=SUBMIT VALUE="Quick Order">
                </FORM>
<!--// end of insertion //-->

<CENTER>$(HorizontalSep)</CENTER>
</TD>
</TR>

</TABLE>
```

```
</CENTER>

%}

  %MESSAGE{100:{%} :continue %}
%}


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

%HTML_REPORT{

<HTML>

<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>$(LongStoreName)</TITLE>
</HEAD>

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

@GET_CATEGORY_BANNERINFO()

<CENTER>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=530>

<TR>
<TD ALIGN="center">
<FONT COLOR="$(TitleTxtCol)" SIZE=3><h2><B>$(CGRYBANNERNAME)</B></H2></FONT>
<HR>
<BR>
</TD>
</TR>

</TABLE>
</CENTER>

@GET_ADDRESS_REF_NUM()


<CENTER>
<TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0>

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

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

</TABLE>
</CENTER>

<BR><BR>

%INCLUDE "/CSPstoremodel/navbar.inc"

</BODY>
</HTML>

%}
```

Appendix B. Net.Data macros for the merchant tool

This appendix contains two Net.Data macros. They are used to create a new feature in the merchant tool.

gftmsg.d2w

```
%{==========================================================================

(C) Copyright IBM Corp. 1999

==========================================================================%}

%define {
    merfnbrVal=merfnbr?"$(merfnbr)":"0"

    giftMessage="0"
    PreGiftText=""

   whereClause = SESSION_ID?",acc_usrgrp,acc_group,shopper where shlogid='$(SESSION_ID)'
and usr_refnum=shrfnbr and refnum=grpu_refnum and mer_rfnbr=merfnbr group by
merfnbr,mestname" : " "

    SHOWSQL="no"
%}


%function(dtw_odbc) checkAccess() {
select shlogid from shopper, acc_usrgrp where shrfnbr = usr_refnum and
mer_rfnbr=$(merfnbrVal) and shlogid = '$(SESSION_ID)'

%REPORT{
%}
%MESSAGE{
100:{
<h2><FONT COLOR="#BB3399">You are not authorized to access the site.</FONT></h2>
%} :exit
%}
%}

%function(DTW_ODBC) StoreName() {
    SELECT DISTINCT merfnbr, mestname
    FROM merchant
    $(whereClause)
    ORDER BY merfnbr, mestname

    %REPORT {
       %ROW{
         @dtw_assign(merfnbrVal,V_merfnbr)
        %}
    %}
%MESSAGE{
100:{ %} :continue
%}
%}

%function(DTW_ODBC) checkGiftMessage() {
    SELECT magftmsg,magfttxt
    FROM maddfeature
    WHERE mamenbr=$(merfnbrVal)
    %REPORT {
       %ROW{
         @dtw_assign(giftMessage,V_magftmsg)
         @dtw_assign(PreGiftText,V_magfttxt)
        %}
    %}
%MESSAGE{
100:{ %} :continue
%}
%}


%HTML_REPORT {

<HTML>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">

<HEAD>
 <TITLE>Gift Message</TITLE>

 <SCRIPT LANGUAGE="JavaScript1.1">
  var ImagePath  = "/csp_gif/";
  var ts_mgrPath = "/cs_mgr/";

  %INCLUDE "ncadmin/formdefs.d2w"
 </SCRIPT>
</HEAD>

<BODY BGCOLOR=#FFFFFF>

<FORM NAME="giftmessage" METHOD="post"
ACTION="/cgi-bin/ncommerce3/ExecMacro/ncadmin/storemgr/gftmsg2.d2w/report">

@StoreName()
@checkAccess()
@checkGiftMessage()

<INPUT TYPE="hidden" NAME="merfnbr" VALUE=$(merfnbrVal)>

<TABLE cols=2 width=520>
<TR>
<TD colspan=2><H2>Gift message from shoppers</H2>
<TR>

<TR>
<TD colspan=2>If enabled then shoppers will be able to enter a gift message
with their order. You should only enable this feature if your business model
allows shoppers to put a gift message on an order.
</TD>
</TR>

<TR>
<TD colspan=2><BR></TD>
</TR>

<TR>
<TD COLSPAN=2>
 %if($(giftMessage) == "1")
  <INPUT CHECKED TYPE="checkbox" NAME="EnableGiftMessage" VALUE="1">Allow shoppers to
enter a gift message
 %else
```

```
  <INPUT TYPE="checkbox" NAME="EnableGiftMessage" VALUE="1">Allow shoppers to enter a
gift message
 %endif
</TR>

<TR>
<TD COLSPAN=2>
Gift message instructions to shoppers
 <TEXTAREA WRAP=PHYSICAL NAME="GiftText" ROWS="3" COLS="50">$(PreGiftText)</TEXTAREA>
</TR>
</TABLE>


<br>
<br>

<INPUT TYPE=submit VALUE="Update">

</FORM>
</BODY>
</HTML>

%}
```

gftmsg2.d2w

```
%{=========================================================================

(C) Copyright IBM Corp. 1999

==========================================================================%}

%define {
    merfnbrVal=merfnbr?"$(merfnbr)":"0"

    giftMessage="0"

    EnableGiftMessageVal=EnableGiftMessage?"1":"0"

   whereClause = SESSION_ID?",acc_usrgrp,acc_group,shopper where shlogid='$(SESSION_ID)'
and usr_refnum=shrfnbr and refnum=grpu_refnum and mer_rfnbr=merfnbr group by
merfnbr,mestname" : " "

    SHOWSQL="no"
%}

%function(dtw_odbc) checkAccess() {
select shlogid from shopper, acc_usrgrp where shrfnbr = usr_refnum and
mer_rfnbr=$(merfnbrVal) and shlogid = '$(SESSION_ID)'

%REPORT{
%}
%MESSAGE{
100:{
<h2><FONT COLOR="#BB3399">You are not authorized to access the site.</FONT></h2>
%} :exit
%}
%}

%function(DTW_ODBC) StoreName() {
    SELECT DISTINCT merfnbr, mestname
    FROM merchant
    $(whereClause)
    ORDER BY merfnbr, mestname

    %REPORT {
       %ROW{
         @dtw_assign(merfnbrVal,V_merfnbr)
        %}
    %}
%MESSAGE{
100:{ %} :continue
%}
%}

%function(DTW_ODBC) checkGiftMessage() {
    SELECT magftmsg
    FROM maddfeature
    WHERE mamenbr=$(merfnbrVal)

    %REPORT {
       %ROW{
         @dtw_assign(giftMessage,V_magftmsg)
        %}
    %}
%MESSAGE{
100:{ %} :continue
%}
%}

%function(DTW_ODBC) insertGiftMessage() {
    INSERT INTO maddfeature
    VALUES ($(merfnbrVal), $(EnableGiftMessageVal), ’$(GiftText)’)

    %REPORT {
       %ROW{
        %}
    %}
%MESSAGE{
100:{    %} :continue
%}
%}

%function(DTW_ODBC) updateGiftMessage() {
    UPDATE maddfeature
    SET magftmsg=$(EnableGiftMessageVal), magfttxt=’$(GiftText)’
    WHERE mamenbr=$(merfnbrVal)

    %REPORT {
       %ROW{
        %}
    %}
%MESSAGE{
100:{ @insertGiftMessage() %} :continue
%}
%}


%HTML_REPORT {

<HTML>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">

<HEAD>
 <TITLE>Gift Message</TITLE>
 <SCRIPT LANGUAGE="JavaScript1.1">
  var ImagePath  = "/csp_gif/";
  var ts_mgrPath = "/cs_mgr/";

  %INCLUDE "ncadmin/formdefs.d2w"
 </SCRIPT>
</HEAD>

<BODY BGCOLOR=#FFFFFF>

@StoreName()
@checkAccess()
@updateGiftMessage()
@checkGiftMessage()

<TABLE cols=2 width=520>
<TR>
<TD colspan=2><H2>Gift message from shoppers</H2></TD>
<TR>

<TR>
<TD colspan=2>
</TD>
</TR>

<TR>
<TD colspan=2><BR></TD>
</TR>

<TR>
<TD COLSPAN=2>
 %if($(giftMessage) == "1")
  Gift message is <b>enabled</b> for all shoppers with the following instructions:<BR>
  $(GiftText)
 %else
  Gift message is <b>disabled</b> for all shoppers.
 %endif
</TD>
</TR>

</TABLE>

<br>
<br>

</BODY>
</HTML>

%}
```

Appendix C. Net.Data macro for the checkout flow

This appendix contains a new version of the ord_pay.d2w macro found in the directory /usr/lpp/NetCommerce3/macro/en_US/CSPstoremodel. It is enhanced to support a gift message function.

ord_pay.d2w

```
%{=====================================================================

(c)  Copyright  IBM Corp.  1999.      All Rights Reserved

=====================================================================%}

%INCLUDE "/CSPstoremodel/translation_text.inc"

%define {
   SHOWSQL="NO"
   TOT_TAXRATE="0"
   TAXRULE_EXISTS="NO"
   SHIPRULE_EXISTS="NO"
   CC_ENABLED="NO"
%}

%function(dtw_odbc) GET_GIFTMESSAGE() {
   SELECT   magftmsg,magfttxt
   FROM     maddfeature
   WHERE    mamenbr=$(MerchantRefNum)

   %REPORT {

      %IF (V_magftmsg == "1")

   <TR>
      <TD>
         <FONT SIZE=3><B>$(TXT_GIFTMESSAGE): </B></FONT><BR>
         $(V_magfttxt)
      </TD>
   </TR>
   <TR>
      <TD>
         <CENTER>
         <TEXTAREA WRAP=PHYSICAL NAME="giftmessage" ROWS="3" COLS="40"></TEXTAREA>
         </CENTER>
      </TD>
   </TR>
   <TR>
      <TD>
         <HR>
      </TD>
   </TR>

      %ENDIF

   %}

  %MESSAGE{
   100:{ %} :continue
   %}
%}
```

```
%function(dtw_odbc) GET_CCMESSAGE() {

   SELECT   pmentinst3
   FROM  merpayinfo
   WHERE    paymerid=$(MerchantRefNum)

   %REPORT {


   %ROW {

      @DTW_assign(CC_MESSAGE, V_pmentinst3)

   %}


   %}

  %MESSAGE{
   100:{ %} :continue
   default:{ error%} :continue
   %}

%}



%function(dtw_odbc) GET_OFFLINE() {

   SELECT   pofflineen, pmentinst1
   FROM  merpayinfo
   WHERE    paymerid=$(MerchantRefNum)

   %REPORT {


   %ROW {

      %IF (V_pofflineen == "1" && CC_ENABLED == "YES")

      <TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0>
         <TR>
            <TD>
               - OR -
            </TD>
         </TR>

         <TR>
            <TD>
               <BR>
            </TD>
         </TR>

      </TABLE>

      %ENDIF


      %IF (V_pofflineen == "1")

      <TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0>
      <TR>
         <TD>
            $(V_pmentinst1)
         </TD>
      </TR>

      <TR>
         <TD>
            <BR>
         </TD>
      </TR>

      </TABLE>

      %ENDIF

      %IF (V_pofflineen == "1" && CC_ENABLED == "YES")

      <TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=8>
         <TR>
            <TD VALIGN="top" COLSPAN=8>
            <input type=radio name=cctype CHECKED VALUE="">$(LBL_PAYOFFLINE)<BR><BR></TD>
         </TR>

      </TABLE>

      %ENDIF

   %}


   %}

  %MESSAGE{
   100:{ %} :continue
   default:{ Error with GET_OFFLINE() %} :continue
   %}

%}


%function(dtw_odbc) GET_CREDITCARDS() {

   select   mccpcard, pmentinst3
   from  mccardinfo, merpayinfo
   where    mccpmerid=$(MerchantRefNum) and paymerid=$(MerchantRefNum)

   %REPORT {

   @DTW_assign(CC_ENABLED, "YES")

   $(CC_MESSAGE)
   <BR><BR>
   <TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0>

   <TABLE WIDTH=200 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=8>

      %ROW {

         <TD VALIGN="top"><input type=radio name=cctype value=$(V_mccpcard)></TD>
         <TD VALIGN="top"><img src="/CHS/images/$(V_mccpcard).gif"></TD>
      %}
      </TR>
   </TABLE>

   <TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0>

      <TR><TD COLSPAN=3><BR></TD></TR>
      <TR>
         <TD ALIGN="left"><FONT SIZE=2>$(LBL_CCNUM)</FONT></TD>
         <TD ALIGN="left"><FONT SIZE=2>$(LBL_CCXMONTH)</FONT></TD>
         <TD ALIGN="left"><FONT SIZE=2>$(LBL_CCXYEAR)</FONT></TD>
      </TR>

      <TR>
         <TD ALIGN="left" VALIGN=middle>
            <INPUT TYPE=text SIZE=15 MAXLENGTH=256 NAME="ccnum" VALUE="$(ccnum)">
         </TD>

         <TD ALIGN="left" VALIGN=middle>
            <select name="ccxmonth" size=1>&nbsp&nbsp
               <option selected></option>
               <option value="1">January</option>
               <option value="2">February</option>
               <option value="3">March</option>
               <option value="4">April</option>
               <option value="5">May</option>
               <option value="6">June</option>
               <option value="7">July</option>
               <option value="8">August</option>
               <option value="9">September</option>
               <option value="10">October</option>
               <option value="11">November</option>
               <option value="12">December</option>
            </select>
         </td>

         <td align="left" valign=middle>
            <select name="ccxyear" size=1>
               <option selected></option>
               <option value="1998">1998 </option>
               <option value="1999">1999</option>
               <option value="2000">2000</option>
               <option value="2001">2001</option>
               <option value="2002">2002</option>
               <option value="2003">2003</option>
               <option value="2004">2004</option>
            </SELECT>
         </TD>

      </TR>

      <TR>
         <TD COLSPAN=3><BR></TD>
      </TR>

   </TABLE>

   %}

  %MESSAGE{
   100:{ %} :continue
   default:{ Error with GET_CREDITCARDS() %} :continue
   %}
```

```
%}


%function(dtw_odbc) SET_TAXRULE_FLAG() {

   SELECT   txrlnbr
   FROM  ptaxrule
   WHERE txmenbr=$(MerchantRefNum)

   %REPORT {

      @DTW_assign(TAXRULE_EXISTS, "YES")
   %}

   %MESSAGE{
      100: {%}:CONTINUE
      default: {ERROR in SET_TAXRULE_FLAG() function in ord_pay.d2w. %}:CONTINUE
   %}

%}


%function(dtw_odbc) SET_SHIPRULE_FLAG() {

   SELECT   sprlnbr
   FROM  pshiprule
   WHERE spmenbr=$(MerchantRefNum)

   %REPORT {

      @DTW_assign(SHIPRULE_EXISTS, "YES")

   %}

   %MESSAGE{
      100: {%}:CONTINUE
      default: {ERROR in SET_SHIPRULE_FLAG() function in ord_pay.d2w. %}:CONTINUE
   %}

%}


%function(dtw_odbc) GET_SHIPPING_INFORMATION() {

   SELECT   smcarrid, smspmode
   FROM  shipmode, shipto
   WHERE stornbr=$(order_rn) and stsmnbr=smrfnbr

   %REPORT {

      @DTW_CONCAT(V_smcarrid, V_smspmode, SHIPRULE_MESSAGE)

   %}

   %MESSAGE{
      100: { %}:CONTINUE
      default: {ERROR in GET_SHIPPING_INFORMATION() function in ord_pay.d2w. %}:CONTINUE
   %}

%}
```

```
%function(dtw_odbc) DISPLAY_DETAILS_LIST() {

   select   strfnbr, stsanbr, stshnbr, stmenbr, stprnbr, stprice, stquant, stcpcur,
            prrfnbr, prldesc2, prsdesc, salname, safname
   from     shipto, product, shaddr
   where    stshnbr=$(SHOPPER_REF) and stmenbr=$(MerchantRefNum) and stprnbr=prrfnbr and
stornbr=$(order_rn)
            and stsanbr=sarfnbr
   order by    stmenbr, stsanbr, strfnbr

   %REPORT{

      <BR>
      <CENTER>
      <TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0 ALIGN="center">

         <TR>
            <TD COLSPAN=3>
            <B><FONT SIZE=2>$(TXT_PURCHASELIST)</FONT></B>
            </TD>
         </TR>
      </TABLE>
      </CENTER>
      <BR>

      <CENTER>
<TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0 ALIGN="center">


         <TR>

             <TD><B><FONT SIZE=2>$(LBL_PRODUCTNUM)</FONT></B></TD>
            <TD><B><FONT SIZE=2>$(LBL_PRODUCTNAME)</FONT></B></TD>
            <TD><B><FONT SIZE=2>$(LBL_QUANTITY)</FONT></B></TD>
             <TD><B><FONT SIZE=2>$(LBL_PRODUCTPRICE)</FONT></B></TD>
             <TD><B><FONT SIZE=2>$(LBL_SUBTOTAL)</FONT></B></TD>

         </TR>


      %ROW{

         <TR>
            @DTW_MULTIPLY(V_stquant, V_stprice, SUB_TOT)

             <TD><FONT SIZE=2>$(V_prldesc2)</FONT></TD>
            <TD><FONT SIZE=2> $(V_prsdesc)</FONT></TD>
            <TD><FONT SIZE=2> $(V_stquant)</FONT></TD>
             <TD><FONT SIZE=2>$(CURRENCY) $(V_stprice) $(V_stcpcur)</FONT></TD>
             <TD BGCOLOR="white" ALIGN="right"><FONT SIZE=2>$(CURRENCY) $(SUB_TOT)
$(V_stcpcur)</FONT></TD>


         </TR>
         <TR><TD HEIGHT=5></TD></TR>

      %}

      </TABLE>
      </CENTER>

   %}
   %MESSAGE{
      100    : {<BR><FONT SIZE=2><B>$(MSG_ORDERLIST_EMPTY)</B></FONT>%}:continue
      default: {ERROR : Problem with DISPLAY_DETAILS_LIST function %} :continue
   %}
%}


%function(dtw_odbc) SHOW_TOTAL_PRICE_MerchantTax() {

   SELECT   distinct orprtot, ortxtot, orshtot, orshtxtot, orcpcur,
         sarfnbr, safname, salname, saaddr1, saaddr2, sacity, sastate, sacntry, sazipc
   FROM  orders, shaddr, shipto
   WHERE ormenbr=$(MerchantRefNum) and orrfnbr=$(order_rn) and stsanbr=sarfnbr and
stornbr=$(order_rn)
   and saadrflg='P'

   %REPORT{

      <TABLE  BORDER=0 CELLSPACING=1 CELLPADDING=1 WIDTH=530 ALIGN="left" COLS=4>


      <TR>
         <TD width=100>
         </TD>
         <TD width=80>
         </TD>
         <TD width=220>
         </TD>

      </TR>


      %ROW {

         @DTW_ADD(V_orprtot, V_ortxtot, total)
         @DTW_ADD(total, V_orshtot, total)
         @DTW_ADD(total, V_orshtxtot, total)

         <TR>
            <TD COLSPAN=4 bgcolor="#E0E0E0">
               $(TXT_ORDERSENTTO)
               <BR><B>$(V_safname) $(V_salname)</B> $(V_saaddr1),
               %IF (V_saaddr2 != "")
               $(V_saaddr2),
               %ENDIF
               $(V_sacity), $(V_sastate), $(V_sacntry), $(V_sazipc)

            </TD>
         </TR>

         <TR><TD COLSPAN=3><BR></TD></TR>

         <TR>
            <TD ALIGN=right VALIGN=top>
               <FONT SIZE=2><B>$(LBL_SUBTOTAL)</B></FONT>
            </TD>
            <TD ALIGN=right VALIGN=middle  bgcolor="white">
               <FONT SIZE=2>$(CURRENCY) $(V_orprtot) $(V_orcpcur) </FONT>
            </TD>
         </TR>
         <TR>
            <TD ALIGN=right VALIGN=top>
               <FONT SIZE=2><B>$(LBL_TAX)</B></FONT>
            </TD>
            <TD ALIGN=right VALIGN=middle BGCOLOR="white">
               %IF (TAXRULE_EXISTS == "YES")
                  <FONT SIZE=2>$(CURRENCY) $(V_ortxtot) $(V_orcpcur)</FONT>
               %ELSE
                  <FONT SIZE=2>-----</FONT>
               %ENDIF
            </TD>
         </TR>



         <TR>
            <TD ALIGN=right VALIGN=top>
               <FONT SIZE=2><B>$(LBL_SHIPPING)</B></FONT>
            </TD>
            <TD ALIGN=right VALIGN=middle BGCOLOR="white">
               %IF (SHIPRULE_EXISTS == "YES")
                  <FONT SIZE=2>$(CURRENCY) $(V_orshtot) $(V_orcpcur)</FONT>
               %ELSE
                  <FONT SIZE=2>-----</FONT>
               %ENDIF

            </TD>


            <TD BGCOLOR="white">
               %IF (SHIPRULE_EXISTS == "YES")
                  <FONT SIZE=2>*$(TXT_SHIPPING_METHOD)*</FONT>
               %ELSE
                  <FONT SIZE=2></FONT>
               %ENDIF
            </TD>
         </TR>

         <TR>

            <TD></TD>
            <TD></TD>
            %IF (ProviderList != "")
               <TD>
               <FONT SIZE=2><B>$(TXT_SHIPPING_OPTIONS)</B></FONT>
               </TD>
            %ENDIF

         </TR>

         <TR>

            <TD></TD>
            <TD></TD>
            %IF (ProviderList != "")
               <TD ALIGN=right VALIGN=middle>
                  <form action="/cgi-bin/ncommerce3/OrderShippingUpdate" method=post>
                  <INPUT TYPE=hidden NAME=addr_rn        VALUE=$(V_sarfnbr)>
                  <INPUT TYPE=hidden NAME=order_rn       VALUE=$(order_rn)>
                  <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderDisplay?status=P&merchant_rn=$(MerchantRefNum)&PAGE=PAYU
PDATE&SHOPPER_REF=$(SHOPPER_REF)&ADDRESS_REF=$(V_sarfnbr)&SHOPPER_TYPE=$(SHOPPER_TYPE)">
```

```
                  <TABLE border=0 width=100% background=$(BodyImage) CELLSPACING=0
CELLPADDING=0 VSPACE=0 HSPACE=0>
                     <TR>
                        <TD>
                           <select name=shipmode_rn>
                              $(ProviderList)
                           </select>
                        </TD>
</TR>
<TR>
                        <TD>
                           <INPUT TYPE=submit VALUE="$(TXT_CHANGE_SHIPPING_SERVICE)">
                        </TD>
                     </TR>
                  </TABLE>
               </TD>
            %ENDIF

         </TR>


          <TR>
            <TD ALIGN=right VALIGN=top>
               <FONT SIZE=2><B>$(TXT_SHIPPINGTAX)</B></FONT>
            </TD>
            <TD ALIGN=right VALIGN=middle  bgcolor="white">
               <FONT SIZE=2>$(CURRENCY) $(V_orshtxtot) $(V_orcpcur)</FONT>
            </TD>
          </TR>

         <TR><TD></TD></TR>

         <TR>
            <TD ALIGN=right VALIGN=top>
               <B>$(LBL_TOTAL)</B>
            </TD>
            <TD ALIGN=right VALIGN=middle BGCOLOR="white">
               <B>$(CURRENCY) $(total) $(V_orcpcur)</B>
            </TD>
         </TR>

         <TR><TD><BR></TD></TR>

      %}

      </TABLE>
      </FORM>

   %}



   %MESSAGE{

      100: {
         <FONT COLOR="Green">SHOW_TOTAL_PRICE_MerchantTax(): Nothing Found</FONT>
         %}:CONTINUE

      default: {
         <FONT COLOR="red">SHOW_TOTAL_PRICE_MerchantTax(): Error</font>
         %}:CONTINUE
   %}
```

```
%}

%function(dtw_odbc) CHECK_ORDER_STATUS() {

   SELECT orstat
   FROM   orders
   WHERE  ormenbr=$(MerchantRefNum) and orrfnbr=$(order_rn)

   %REPORT {
      %ROW {
               @DTW_assign(ORDER_STATUS, V_orstat)
      %}
   %}

   %MESSAGE{
      100:{ %} :continue
      default:{ Error with CHECK_ORDER_STATUS() %}
   %}
%}

%FUNCTION(dtw_odbc) GetAssignedShipModeRef(){

     SELECT distinct stsmnbr
     FROM  shipto
     WHERE stornbr=$(order_rn) and stsmnbr is not null

     %REPORT{

         %ROW{

              @DTW_assign(AssignedShipRef, V_stsmnbr)
         %}
     %}
     %MESSAGE {
        100: {
              @DTW_assign(AssignedShipRef, "null")
        %}:continue
      default:{ Error with GetAssignedShipModeRef() %}
     %}

%}


%FUNCTION(dtw_odbc) GetFirstShipModeRef(){

     SELECT distinct mmrfnbr, smcarrid, smspmode
     FROM   shipmode, mshipmode
     WHERE  smrfnbr in ( select mmsmnbr from mshipmode where mmmenbr= $(MerchantRefNum) )
            and mmmenbr=$(MerchantRefNum) and mmsmnbr=smrfnbr

     %REPORT{

         @DTW_assign(counter,      "1")
         @DTW_assign(firstShipRef, "null")

         %ROW{
              %IF ( counter == "1" )
                 @DTW_assign(firstShipRef, V_mmrfnbr)
              %ENDIF

              @DTW_add(counter, "1", counter)
         %}
     %}
     %MESSAGE {
        100: {
              @DTW_assign(firstShipRef, "null")
        %}:continue
     %}

%}


%{====================================================%}
%{ HTML Input Section
%{====================================================%}

%HTML_INPUT{

<HTML>

<HEAD>
  <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</head>

<TITLE>$(LongStoreName) [$(TXT_TITLE_BILLTO)]</TITLE>

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

@GetAssignedShipModeRef()
@GetFirstShipModeRef()

%IF (firstShipRef != "null" && AssignedShipRef != "null")

<META HTTP-EQUIV="Refresh"

CONTENT="0;URL=/cgi-bin/ncommerce3/OrderShippingUpdate?order_rn=$(order_rn)&shipmode_rn=
$(AssignedShipRef)&merchant_rn=$(MerchantRefNum)&url=/cgi-bin/ncommerce3/OrderDisplay?st
atus%3dP%26PAGE%3dPAYUPDATE%26SHOPPER_REF%3d$(SHOPPER_REF)%26ADDRESS_REF%3d$(ADDRESS_REF
)%26SHOPPER_TYPE%3d$(SHOPPER_TYPE)">

%ELIF (firstShipRef != "null" && AssignedShipRef == "null")

<META HTTP-EQUIV="Refresh"

CONTENT="0;URL=/cgi-bin/ncommerce3/OrderShippingUpdate?order_rn=$(order_rn)&shipmode_rn=
$(firstShipRef)&merchant_rn=$(MerchantRefNum)&url=/cgi-bin/ncommerce3/OrderDisplay?statu
s%3dP%26PAGE%3dPAYUPDATE%26SHOPPER_REF%3d$(SHOPPER_REF)%26ADDRESS_REF%3d$(ADDRESS_REF)%2
6SHOPPER_TYPE%3d$(SHOPPER_TYPE)">

%else

<META HTTP-EQUIV="Refresh"

CONTENT="0;URL=/cgi-bin/ncommerce3/ExecMacro/$(DirectoryName)/ord_pay.d2w/report?ADDRESS
_REF=$(ADDRESS_REF)&SHOPPER_REF=$(SHOPPER_REF)&order_rn=$(order_rn)">

%ENDIF

</BODY>
</HTML>
%}


%{====================================================%}
```

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

%HTML_REPORT {

<HEAD>
   <META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
   <TITLE>$(LongStoreName) [$(TXT_TITLE_PAYMENT)]</TITLE>
</HEAD>


<HTML>

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


<!-- Debug Information
<FONT COLOR=purple>
<BR><B>Environment Information</B>
<BR>order_rn =  $(order_rn)
<BR>SHOPPER_REF =  $(SHOPPER_REF)
<BR>ADDRESS_REF =  $(ADDRESS_REF)
<BR><B>Query Information</B>
</FONT>
<BR><BR>
-->

<CENTER>
<TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0 ALIGN="center">

   <TR>
      <TD ALIGN="center" VALIGN="center">
         <FONT FACE="helvetica" COLOR=$(TitleTxtCol)><H3>$(TXT_TITLE_PAYMENT)</H3></FONT>
      </TD>
   </TR>

   <TR>
      <TD>
         <FONT SIZE=2>$(TXT_PAGEDESCRIPTION_PAYMENT)
         <BR>
      </TD>
   </TR>

</TABLE>
</CENTER>

@SET_TAXRULE_FLAG()
@SET_SHIPRULE_FLAG()
@GET_SHIPPING_INFORMATION()
@CHECK_ORDER_STATUS()


<CENTER>
<TABLE WIDTH=530 CELLPADDING=0 CELLSPACING=0 BORDER=0 ALIGN="center">

   <TR>
      <TD COLSPAN=2>@SHOW_TOTAL_PRICE_MerchantTax()</TD>
   </TR>

   <TR>
      <TD COLSPAN=2>@DISPLAY_DETAILS_LIST()</TD>
   </TR>
   <TR>
      <TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=CENTER></TD>
   </TR>

   %IF (ORDER_STATUS != "C")

      <FORM ACTION="/cgi-bin/ncommerce3/OrderProcess" METHOD="post">
      @GET_GIFTMESSAGE()

   <TR>
      <TD>
         <FONT SIZE=3><B>$(TXT_PAYMENT):</B></FONT>
      </TD>
   </TR>

   <TR>
      <TD>
         @GET_CCMESSAGE()
         @GET_CREDITCARDS()
         @GET_OFFLINE()

      </TD>
   </TR>

   <TR>
      <TD ALIGN="center">
         <BR>
         <INPUT TYPE=hidden NAME=merchant_rn    VALUE=$(MerchantRefNum)>
         <INPUT TYPE=hidden NAME=order_rn       VALUE=$(order_rn)>
         <INPUT TYPE=hidden NAME=billto_rn       VALUE=$(ADDRESS_REF)>
         <INPUT TYPE=hidden NAME=ADDRESS_REF    VALUE=$(ADDRESS_REF)>
 <INPUT TYPE=hidden NAME=SHOPPER_REF    VALUE=$(SHOPPER_REF)>
         <INPUT TYPE=hidden NAME=TAXRULE_EXISTS    VALUE=$(TAXRULE_EXISTS)>
         <INPUT TYPE=hidden NAME=SHIPRULE_EXISTS   VALUE=$(SHIPRULE_EXISTS)>
         <input type=submit value="$(BUT_PURCHASE)">
         </FORM>
      </TD>
   </TR>
   %ELSE
   <TR>
      <TD COLSPAN=2 ALIGN="center">
         <b>$(TXT_ORDER_COMPLETE)</b><br><br>
         <A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgmenbr=$(MerchantRefNum)&cgrfnbr=$(HomeCatego
ryNum)"><B>$(LINK_CONTINUE)</B></A>
      </TD>
   </TR>

   <TR><TD><BR></TD></TR>

   %ENDIF

</TABLE>
</CENTER>

%INCLUDE "/CSPstoremodel/navbar.inc"

</BODY>

</HTML>

%}
```

Appendix D. Source code for AddGiftMessage OF

This appendix contains the makefile and the source code for the AddGiftMessage overridable function (OF).

makefile.giftmessage

```
##
==========================================================================
=====
##
## (c)  Copyright  IBM Corp. 1999.      All Rights Reserved
##
##
##
==========================================================================
=====
##
##   - Run the makfile as follows:
##
##       make -f makefile.giftmessage
##

NCROOT          = /usr/lpp/NetCommerce3
TARNAME         = addgiftmessage
OUTDIR          = $(NCROOT)/bin
OUT             = lib$(TARNAME).a

CC              = xlC_r
LINKSHR         = /usr/ibmcxx/bin/makeC++SharedLib_r
CFLAGS          = $(DEBUG) -DAIX -qlistopt -qlist -qxref=full -c
GCAPILIBFLAGS   = -+ $(DEBUG) -DAIX -DNLS_ENABLED -DCLNK
INC             = -I$(NCROOT)/adt/include
LIBS            = -L. -L$(NCROOT)/bin \
                  -lnc3_containers    \
                  -lnc3_messages      \
                  -lnc3_common        \
                  -lnc3_dbc           \
                  -lserver_objs

OBJS             = tasks_api.o        \
                   addGiftMessage.o

all             : $(OUT)

$(OUT)          : $(OBJS)
```

```
  $(LINKSHR) -bloadmap:api.loadmap -p 0 \
-o $@ $(OBJS) $(LIBS)

addGiftMessage.o : addGiftMessage.cpp
$(CC) $(INC) $(CFLAGS) addGiftMessage.cpp

tasks_api.o:  tasks_api.cpp
   $(CC) $(INC) $(CFLAGS) tasks_api.cpp

clean          :
rm *.o *.lst *.loadmap
rm $(OUT)
```

addGiftMessage.cpp

```
//
------------------------------------------------------------------------
---------
//
// (c)  Copyright  IBM Corp. 1999          All Rights Reserved
//
//
// FILE NAME:    addGiftMessage.cpp
//
// DESCRIPTION:  Net.Commerce overridable function for the EXT_ORD_PROC
task.
//
//
------------------------------------------------------------------------
---------

#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

// -----------------------------------------------------------------
// Uncomment the #define statement below to enable the trace
// facilities.
// -----------------------------------------------------------------
// #define __TRACE_NCAPIS__

#ifdef __TRACE_NCAPIS__
  typedef TraceYes Trace;
#else
  typedef TraceNo  Trace;
#endif
static Trace trace("NC_APIS ("__FILE__")");


// ------------------------------------------------------------------
// The overridable function addGiftMessage
// ------------------------------------------------------------------

class __DLL_NCS_API__ addGiftMessage : public NC_OverridableFunction {
   static const ClassName _STR_ThisClass;

   public:
     addGiftMessage() {
       Trace::Tracer T(_STR_CONSTRUCTOR, _STR_ThisClass);
     }
     virtual ~addGiftMessage() {
       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(_STR_Process, _STR_ThisClass);

       return ProcessGiftMessage(Req, Res, Env);
     }


     virtual bool ProcessGiftMessage(const HttpRequest& Req, HttpResponse&
Res, NC_Environment& Env) {
       Trace::Tracer T("ProcessGiftMessage", _STR_ThisClass);

       // ----------------------------------------------------------
       // This parameter ORDER_REF_NUN is documented by Net.Commerce
       // as part of the interface between the command and the API.
       // ----------------------------------------------------------
       const String& OrderRefNum    = *(const String *)
Env.Seek("ORDER_REF_NUM");
```

```
long ordernumber = 0;
long merchantnumber = 0;

OrderRefNum.getVal(ordernumber);
_MerchantRefNum.getVal(merchantnumber);

// --------------------------------------------------------
// Get the name value pair 'giftmessage' from the URL posted
// by the browser.
// --------------------------------------------------------

const StringWithOwnership NVP_giftmessage("giftmessage");

const NameValuePairMap& NVPMap = Req.getNVPs();
const String& GiftMessage = NVPMap.Get(NVP_giftmessage);

// --------------------------------------------------------
// Check if the 'giftmessage' parameter has been specified
// --------------------------------------------------------
if ( GiftMessage.IsEmpty() ) {
 trace << indent << " ProcessGiftMessage : No Gift Message"
        << " for order ORRFNBR = " << ordernumber << endl;
 return true;
}

// --------------------------------------------------------
// Add the gift message to order in the ORDERS table.
// --------------------------------------------------------

String Stmt;
DataBase *DbConnection = DataBaseManager::GetCurrentDataBase();

Stmt.Clean();
Stmt << "UPDATE ORDERS "
     << "SET ORFIELD3 = '" << GiftMessage << "' "
     << "WHERE ORRFNBR = " << ordernumber;

SQL SqlUpdateGiftMessage(*DbConnection, Stmt);

if (SqlUpdateGiftMessage.getSQLrc() != ERR_DB_NO_ERROR) {
  SqlUpdateGiftMessage.ReportError();
  trace << indent << " ProcessGiftMessage : Updating gift"
        << " message failed. ORRFNBR = " << ordernumber << endl;
}
else {
  trace << indent << " ProcessGiftMessage : Updating gift"
        << " message for order ORRFNBR = " << ordernumber << endl;
```

```
       }

       // --------------------------------------------------------
       // We always return TRUE even when something fails or the
       // database would be rolled back (= no order).
       // --------------------------------------------------------
       return true;
     }
};


const ClassName addGiftMessage::_STR_ThisClass("AddGiftMessage");

static bool X =
NC_OverridableFunctionManager::GetUniqueInstance().RegisterApi
                ( "IBM ITSO", "NC", "AddGiftMessage", 1.0, new
addGiftMessage );
```

Appendix E. Template file for the order details page

This appendix contains customized version of the InvoicePage.tem template file found in /usr/lpp/NetCommerce3/CHS/mpg_templates directory. It is used to customize **manage orders** menu in the merchant tool for managing gift message.

InvoicePage.tem

```
main()
{
  Sub env, resources, data

  Var selectedType

  selectedType = ""

/*

<!--==========================================================================
IBM Confidential

OCO Source Materials

5648-B47

(C) Copyright IBM Corp. 1998

The source code for this program is not published or otherwise divested of its
trade secrets, irrespective of what has been deposited with the U.S. Copyright
Office.

==========================================================================-->
<HTML>
<HEAD>
   <BASE HREF="https://$env.hostname$/">
   <META HTTP-EQUIV="expires" CONTENT="0">
   <META NAME="GENERATOR" CONTENT="Mozilla/4.04 [en] (WinNT; U) [Netscape]">
   <TITLE>invoice</TITLE>
<SCRIPT>

function openHelp(fileName) {

     a = window.open("http://$env.hostname$/nchelp/panelinf/" + fileName, 'help',
'toolbar=no,menubar=yes,width=540,height=480');

  }

function showNumber(order_rn) {

    a = window.open("https://$env.hostname$/cgi-bin/ncommerce3/GetPaymentInfo?order_rn="
+ order_rn + "&url=/cgi-bin/ncommerce3/ExecMacro/pdi.d2w/report" , 'view',
'toolbar=no,menbar=yes,width=300,height=250');
    a.focus()
}


function reallyRemove() {
  return  confirm("$resources.RemoveConfirm$ #$data.orderNumber$
$resources.QuestionMark$");
}

function getOrderType(sType) {

    //alert("$data.SelectedType$");
    //var type = '$data.SelectedType$';
    //var type = '$selectedType$';
    var type = sType;

    if ( type == '$resources.PENDING_STATE$' )
      document.changeStatus.SelectedType.value = "0";
    else if ( type == '$resources.COMPLETE_STATE$' )
            document.changeStatus.SelectedType.value = "1";
         else if ( type == '$resources.FAILED_STATE$' )
                    document.changeStatus.SelectedType.value = "8";
              else if ( type == '$resources.INVENTORY_STATE$' )
                     document.changeStatus.SelectedType.value = "3";
                   else if ( type == '$resources.CANCEL_STATE$' )
                           document.changeStatus.SelectedType.value = "4";
                        else if ( type == '$resources.SHIPPED_STATE$' )
                               document.changeStatus.SelectedType.value = "5";
                             else if ( type == '$resources.AUTHORIZED_STATE$' )
                                     document.changeStatus.SelectedType.value = "6";
                                  else if ( type == '$resources.NEWORDER_STATE$' )
                                         document.changeStatus.SelectedType.value = "7";
                                       else
                                         document.changeStatus.SelectedType.value = "0";

}

</SCRIPT>

</HEAD>
<BODY BGCOLOR='WHITE'>

<!--==========================================================================
   Qeury to retrieve the product, order, address, tax, shipto , shipping and
   payment info.

==========================================================================-->
*/

   -- --------------------------------------------------- --
   -- Check this is a cybercash failed                    --
   -- --------------------------------------------------- --
    Var           cyberFailed
    Var           taxDescription
    cyberFailed = "false"
    taxDescription = resources.TAXDESCRIPTION + ":"
    Query  stmt20

    stmt20        = "select orrfnbr from orders, ordpaymthd, cypaymthd " +
                    "where orrfnbr=" + data.selectedOrder +
                    " and orrfnbr=omornbr and ompaymthd='CYBER' and " +
                    "cyornbr=orrfnbr and cystatus = 'failure-hard' "
    repeat ( stmt20 ) {

        cyberFailed = "true"
    }
    Query   stmt1
    Var     lastUpdate, billToRef, shipto, billto, itemList
    Var     satitle, prldesc1, prldesc2, stfield2, saaddr2, saaddr3
    Var     currency, taxCode, country, state

    saaddr2= ""
    saaddr3= ""

   -- ---------------------------------------------------- --
   -- Section for Order Information:                         --
   --                                                        --
   -- Table Used: shipto, product, shaddr, orders            --
   --                                                        --
   -- ---------------------------------------------------- --
    stmt1= "select satitle, salname, safname, samname, saphone1, saaddr1, " +
           "saaddr2, saaddr3, sacity, sastate, sacntry, sazipc,  strfnbr, " +
           "stornbr, stsanbr, stshnbr, stprnbr,stprice, stcpcur, stpcode, " +
           "stquant, prsdesc, prldesc2, orbllto, orustmp, stcmt " +
           "from shipto, product, shaddr , orders " +
           "where stornbr=" + data.selectedOrder +
           " and prrfnbr=stprnbr and sarfnbr=stsanbr and orrfnbr= stornbr "

    itemList = ""

    repeat (stmt1) {

        lastUpdate =  stmt1.orustmp    -- last update date for current order

        if ( stmt1.satitle == "null" ) satitle = ""    -- no title for this shopper
        else satitle = stmt1.satitle

       -- if ( stmt1.prldesc1== "null" ) prldesc1= ""    -- no 1st description field for
this product
        -- else prldesc1= stmt1.prldesc1
        prldesc1 = ""

        if ( stmt1.prldesc2== "null" ) prldesc2= ""    -- no 2nd description field for
this product
        else prldesc2= stmt1.prldesc2

        if ( stmt1.stcmt   == "null" ) stfield2= ""       -- no comment field for this
product
        else stfield2= stmt1.stcmt

        if ( stmt1.saaddr2 == "null" ) saaddr2 = ""    -- no 2nd address field for this
shopper
        else saaddr2 = stmt1.saaddr2

        if ( stmt1.saaddr3 == "null" ) saaddr3 = ""    -- no 3rd addres field for this
shopper
        else saaddr3 = stmt1.saaddr3

       -- the list of orders for a store, this will be imbeded into a select box in html
page --

        if ( cyberFailed == "true" )
          selectedType = resources.FAILED_STATE
        else
          selectedType = data.SelectedType

       itemList += "<TR><TD VALIGN=TOP WIDTH='80'>&nbsp;&nbsp;&nbsp; " + stmt1.stquant +
"</TD>" +
                    "<TD VALIGN=TOP WIDTH='80'>" + stmt1.prsdesc + "</TD>" +
                   "<TD VALIGN=TOP WIDTH='120'>" +  prldesc2 + "<BR><FONT COLOR='red'>"
+ stfield2 + "</FONT></TD>" +
                    "<TD VALIGN=TOP WIDTH='2'>&nbsp;</TD>" +
                    "<TD VALIGN=TOP WIDTH='80'>" + stmt1.stprice + "</TD>" +
                    "<TD VALIGN=TOP WIDTH='120'>" + selectedType + "</TD>" +
                   "<TD VALIGN=TOP>" + stmt1.stprice|multiply(stmt1.stquant)|round(2) +
"</TD></TR>"  -- stprice * stquant

        billToRef = stmt1.orbllto     -- the reference number for bill to address

       -- this is the shipto address including last name, first name, address, telephone
etc. --
        -- It will be imbeded into a html table --
        shipto    = "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + satitle + " " +
stmt1.salname + ", " + stmt1.safname + "</FONT></TD></TR>" +
                   "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt1.saaddr1 + ", " +
saaddr2 + " " + saaddr3 + "</FONT></TD></TR>" +
                    "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt1.sacity + ", " +
stmt1.sastate + "</FONT></TD></TR>" +
                   "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt1.sacntry + ", " +
stmt1.sazipc  + "</FONT></TD></TR>" +
                    "<TR><TD>&nbsp;</TD><TD>&nbsp;</TD></TR>" +
                    "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt1.saphone1 +
"</FONT></TD></TR>"


         --
---------------------------------------------------------------------------------- --
         -- Get tax code description associated the current tax country and state
jurisdictions --

--------------------------------------------------------------------------------------
-
         taxCode   = stmt1.stpcode    -- tax code assigned
         country   = stmt1.sacntry    -- shipto country
         state     = stmt1.sastate    -- shipto state

         Query stmt2
         Var   taxCodeField

         if ( taxCode == "null" ) taxCodeField = " TXPCODE is NULL "                  --
null tax Code
         else                      taxCodeField = " TXPCODE='" + taxCode + "' "

         stmt2 = "select TXCODDESC from taxrule where txmenbr=" + data.Menbr +
                 " and UCASE(txcntry)= '" +  country | upperCase() +
                 "' and UCASE(txstate) = '" +  state | upperCase() +
                 "' and " + taxCodeField

        stmt2|reset()

        if (stmt2|size() != 0) {                   -- found tax code description
           taxDescription +=  stmt2.txcoddesc + "; "
         }
        else {                                    -- cannot find a tax code description,
search for the state is NULL
                Query    stmt3
                stmt3   =  "select TXCODDESC from taxrule where txmenbr=" +
                           data.Menbr + " and UCASE(txcntry)= '" +
                           country | upperCase()  + "' and txstate is NULL and " +
                           taxCodeField
                stmt3|reset()
                if ( stmt3|size() != 0 ) {          -- found tax code description
                    taxDescription += stmt3.txcoddesc + "; "
                }
                else  taxDescription = ""
          }

     }

        currency  = stmt1.stcpcur    -- what currency ?
        taxCode   = stmt1.stpcode    -- tax code assigned
        country   = stmt1.sacntry    -- shipto country
        state     = stmt1.sastate    -- shipto state


   -- ------------------------------------------------------------------ --
   -- Section: get shipping mode and services                            --
   --                                                                    --
   -- ------------------------------------------------------------------ --
   Var   shippingMode
   Query stmt4

   stmt4 = "select distinct smcarrid, smspmode, smspdesc from shipmode, " +
            "mshipmode, product, shipto where smrfnbr=mmsmnbr and mmrfnbr " +
            "in (select stsmnbr from shipto where stornbr=" + data.selectedOrder +
            " ) and stprnbr=prrfnbr "

   stmt4|reset()
   if ( stmt4|size() != 0 ) {
        shippingMode = stmt4.smcarrid + "," + stmt4.smspmode
     }
   else shippingMode = ""

   -- ----------------------------------------------------------------- --
   -- Bill to Infomation                                                --
   -- ----------------------------------------------------------------- --
   Var safname

   if ( billToRef == "null" ) {  --  bill-to is the same as shipto
      billto = resources.THE_SAME_AS_BILLTO
   }
   else { -- get bill-to info
     Query stmt5

     stmt5 ="select satitle, salname, safname, samname, saphone1, saaddr1, saaddr2, " +
            " saaddr3, sacity, sastate, sacntry, sazipc " +
            " from shaddr where sarfnbr=" + billToRef

     stmt5|reset()
     if ( stmt5.satitle == "null" ) satitle = " "
     else satitle = stmt5.satitle
     if ( stmt5.safname == "null" ) safname = " "
     else safname = stmt5.safname
     if ( stmt5.saaddr3 == "null" ) saaddr3 = " "
     else saaddr3 = stmt5.saaddr3
     if ( stmt5.saaddr2 == "null" ) saaddr2 = " "
     else saaddr2 = stmt5.saaddr2

    billto = "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + satitle + " " + stmt5.salname
+ ", " + safname + "</FONT></TD></TR>" + -- name
              "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt5.saaddr1 + ", " +
saaddr2 + " " + saaddr3 + "</FONT></TD></TR>" + -- address
```

```
              "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt5.sacity + ", " +
stmt5.sastate + "</FONT></TD></TR>" + -- country, state
              "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt5.sacntry + ", " +
stmt5.sazipc + "</FONT></TD></TR>" + -- zipcode
              "<TR><TD>&nbsp;</TD><TD>&nbsp;</TD></TR>" +
              "<TR><TD WIDTH='20'></TD><TD><FONT SIZE=-1>" + stmt5.saphone1 +
"</FONT></TD></TR>"


   }

   -- ------------------------------------------------------------------- --
   -- Get tax, shipping , tax on shipping and total charge                 --
   -- ------------------------------------------------------------------- --
   Var total, tax, shipping , taxOnShipping
   Var total_after

   Query stmt6

   stmt6 = "select orprtot, ortxtot, orshtot, orshtxtot from orders where orrfnbr=" +
           data.selectedOrder

   stmt6 |reset()

   if ( stmt6.orprtot   == "null" ) total         = "0.00"
   else                             total         = stmt6.orprtot|floor(2)
   if ( stmt6.ortxtot   == "null" ) tax           = "0.00"
   else                             tax           = stmt6.ortxtot|floor(2)
   if ( stmt6.orshtot   == "null" ) shipping      = "0.00"
   else                             shipping      = stmt6.orshtot|floor(2)
   if ( stmt6.orshtxtot == "null" ) taxOnShipping = "0.00"
   else                             taxOnShipping = stmt6.orshtxtot|floor(2)

   -- get total revenue
   --
   total_after = total|add(tax)|add(shipping)|add(taxOnShipping)|floor(2)

   -- ------------------------------------------------------------------- --
   -- Get CyberCash payment info (online, offline)                        --
   -- ------------------------------------------------------------------- --
  Query stmt7

  Var paymethod, creditType, cardNumber, expireDate, cyberCashState, cyberCashStatus

  creditType = ""
  cardNumber = ""
  expireDate = ""
  cyberCashStatus = ""
  cyberCashState = ""
  paymethod = ""


  stmt7 = "select ompaymthd, ompaydevc, omdeffs from ordpaymthd where omornbr=" +
          data.selectedOrder

  stmt7 |reset()


  if ( stmt7 |size() != "0" ) {


    creditType = stmt7.ompaymthd
    cardNumber = stmt7.ompaydevc
    expireDate = stmt7.omdeffs

    if ( stmt7.ompaymthd == "CYBER" ) { -- Paid by cyberCash
      paymethod = resources.CYBERCASH_NAME        -- the name of cybercash
      Query stmt8
      stmt8 = "select cystate, cystatus, cycard_number, cycard_exp from cypaymthd where
cyornbr=" +
               data.selectedOrder
      stmt8 |reset()
      if ( stmt8 | size() != "0" ) {
          if ( stmt8.cycard_number | isDigit() == "true" )
             cardNumber = stmt8.cycard_number
          else
             cardNumber = " "

          cyberCashState = resources.CYBERCASH_STATE + " : " +   stmt8.cystate
          cyberCashStatus= resources.CYBERCASH_STATUS + " : " +  stmt8.cystatus
          expireDate     = stmt8.cycard_exp
        }
     }
    else { -- non CyberCash Payment
       if ( stmt7.ompaydevc |substring(1,8) == "OFF-LINE" ) { -- off line non cyberCash
           paymethod = resources.OFFLINE_PAYMENT
           expireDate= " "
        }
       else {
         paymethod = resources.OFFLINE_PAYMENT
        }

    }

  }

  -- ------------------------------------------------------------------ --
  -- Get Order Remark Info                                               --
  -- ------------------------------------------------------------------ --
  Query stmt9
  Var   history

  history = ""

  stmt9 = "select ortmstmp, orcoment from orcomment where orrefnbr=" +
          data.selectedOrder + " order by ortmstmp "

  repeat ( stmt9 ) {
         history += "\n[" + stmt9.ortmstmp + "]\n" + stmt9.orcoment + "\n\n"
  }


  -- ------------------------------------------------------------------ --
  -- Get Gift Message                                                    --
  -- inserted for Gift Message customization                             --
  -- ------------------------------------------------------------------ --
  Query stmt10
  Var   giftmsg

  giftmsg = ""

  stmt10 = "select orfield3 from orders where orrfnbr=" +
          data.selectedOrder

  stmt10 | reset()
  giftmsg += stmt10.orfield3
```

```
/*

<!--=============================================================================
   End of Qeury
=============================================================================-->

<H2>$resources.INVOICE_PAGE$</H2>
<BR>
<TABLE BORDER=0  COLS=1 WIDTH="602" BGCOLOR="#6699CC" >
   <TR>
     <TD><B>$resources.OrderInfo$</B></TD>
   </TR>
</TABLE>

<TABLE>
   <TR>
       <TD>&nbsp;</TD>
  </TR>
  <TR>
     <TD>$resources.OrderNumber$&nbsp;$data.orderNumber$</TD>
     <TD>&nbsp;</TD>
     <TD>$resources.LastUpdate$&nbsp;$lastUpdate$</TD>
   </TR>
   <TR>
       <TD>&nbsp;</TD>
   </TR>
</TABLE>

<TABLE BORDER=0 CELLPADDING=0  CELLSPACING=0 CELLPADDING=0  COLS=7 WIDTH="602">
  <TR  BGCOLOR="#CCCCCC">
     <TD WIDTH="80"><B><FONT COLOR="#000099">$resources.Quantity$</FONT></B></TD>
     <TD WIDTH="80"><B><FONT COLOR="#000099">$resources.InvoiceProduct$</FONT></B></TD>
     <TD WIDTH="120"><B><FONT COLOR="#000099">$resources.ProductDesc$</FONT></B></TD>
     <TD WIDTH="2">&nbsp;</TD>
     <TD WIDTH="80"><B><FONT COLOR="#000099">$resources.ProductPrice$</FONT></B></TD>
     <TD WIDTH="120"><B><FONT
COLOR="#000099">$resources.ProductOrderState$</FONT></B></TD>
     <TD><B><FONT COLOR="#000099">$resources.ProductTotal$</FONT></B></TD>
  </TR>

<TR>

$itemList$

</TABLE>

<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0 COLS=4 WIDTH="602" BGCOLOR="#CCCCCC" >
   <TR>
     <TD WIDTH="155">&nbsp;</TD>
     <TD WIDTH="220">&nbsp;</TD>
     <TD WIDTH="140">&nbsp;</TD>
     <TD>&nbsp;</TD>
   </TR>
   <TR>
       <TD WIDTH="155">&nbsp;</TD>
       <TD WIDTH="220">&nbsp;</TD>
       <TD WIDTH="140"><B><FONT COLOR="#000099">$resources.ProductTax$</FONT></B></TD>
       <TD>$tax$</TD>
   </TR>
   <TR>
       <TD WIDTH="155">&nbsp;</TD>
       <TD WIDTH="220">&nbsp;</TD>
       <TD WIDTH="140"><B><FONT
COLOR="#000099">$resources.ProductShipping$</FONT></B></TD>
       <TD>$shipping$</TD>
   </TR>
    <TR>
       <TD WIDTH="155">&nbsp;</TD>
       <TD WIDTH="220">&nbsp;</TD>
       <TD WIDTH="140"><B><FONT
COLOR="#000099">$resources.ProductTaxOnShipping$</FONT></B></TD>
       <TD>$taxOnShipping$</TD>
   </TR>
   <TR>
       <TD WIDTH="155">&nbsp;</TD>
       <TD WIDTH="220">&nbsp;</TD>
       <TD WIDTH="140"><B><FONT COLOR="#000099">______________</FONT></B></TD>
       <TD>____</TD>
   </TR>
   <TR>
       <TD WIDTH="155">&nbsp;</TD>
       <TD WIDTH="220">&nbsp;</TD>
       <TD WIDTH="140"><B><FONT COLOR="#000099">$resources.TotalForOrder$
($currency$)</FONT></B></TD>
       <TD>$total_after$</TD>
   </TR>
   <TR>
     <TD WIDTH="155">&nbsp;</TD>
     <TD WIDTH="220">&nbsp;</TD>
     <TD WIDTH="140">&nbsp;</TD>
     <TD>&nbsp;</TD>
   </TR>
</TABLE>


$taxDescription$

&nbsp;
<BR><BR>

$resources.ShipMode$&nbsp;<FONT COLOR='#000099'>$shippingMode$</FONT>
<BR>


<TABLE BORDER=0  COLS=1 WIDTH="602" BGCOLOR="#6699CC" >
   <TR>
     <TD><b>$resources.AddressInfo$</b></TD>
   </TR>
</TABLE>


<TABLE  COLS=3 WIDTH="540">
    <TR>
       <TD>&nbsp;</TD>
    </TR>

    <TR>
     <TD WIDTH= "250">$resources.ShipTo$
        <TABLE>
           <TR>
              <TD WIDTH="20"></TD>
              <TD></TD>
           </TR>
```

```
             $shipto$
        </TABLE>
     </TD>

   <TD WIDTH="20">&nbsp;

   </TD>

   <TD WIDTH="250">$resources.BillTo$
      <TABLE>
          <TR>
            <TD WIDTH="20"></TD>
            <TD></TD>
          </TR>

         $billto$

     </TABLE>

   </TD>
  </TR>
</TABLE>

<BR>


<TABLE BORDER=0  COLS=1 WIDTH="602" BGCOLOR="#6699CC" >
   <TR>
     <TD><B>$resources.ProductPaymentInfo$</B></TD>
   </TR>
</TABLE>

<TABLE>
    <TR>
    <TD WIDTH="130">$resources.ProductPaymentMethod$</TD>
         <TD>$paymethod$</TD>
    </TR>
    <TR>
         <TD WIDTH="130">$resources.ProductCreditCard$</TD>
         <TD>$creditType$</TD>
    <TR>
         <TD WIDTH="130">$resources.ProductCreditCardNumber$</TD>
         <TD>*/
   if ( cardNumber | strip() | isDigit() == "true" || cardNumber == " " ) {
        /*$cardNumber$*/
    }
   else  if (  cardNumber|occurencesOf("OFF-LINE")  == "1"  )  {
              /*$resources.NO_CREID_CARD_AVA$*/
         }
        else {
            /*

                <FORM NAME="openData">
                   <INPUT TYPE='button' VALUE='View Credit Card Number'
onClick='showNumber($data.selectedOrder$)'>
                </FORM>
           */
        }
/*
        </TD>
     </TR>
     <TR>
        <TD WIDTH="130">$resources.ProductExpireDate$</TD>
        <TD>$expireDate$</TD>
     </TR>
</TABLE>


<BR>&nbsp;

<TABLE>
  <TR><TD>$cyberCashState$</TD></TR>
  <TR><TD>$cyberCashStatus$</TD></TR>
</TABLE>


*/
   addRemoveComments(history, data.runCommand)

   -- inserted for Gift Message customization
   addGiftMessage(giftmsg)

/*

<CENTER></CENTER>

<HR WIDTH="100%">

<CENTER>

<TABLE>
 <TR>
    <TD>
        */
           addChangeStatusButton(data.SelectedType, data.runCommand, selectedType)
        /*
    </TD>
    <TD WIDTH='5'> </TD>
    <TD>
        <FORM NAME='InvoiceCancel' METHOD='POST' ACTION='servlet/MerchantAdmin'>

           <!-- Hidden field to make sure the browser does not cache the confirm page -->
           <SCRIPT>
              var now = new Date();
              document.writeln('<INPUT TYPE=hidden NAME="CTS"  VALUE= "' + now.getTime()
+ '">');
           </SCRIPT>


           <INPUT TYPE='hidden' NAME='GOTO' VALUE='CHS_OrderProcess'>
           <INPUT TYPE='submit' NAME='ToProcessPage'
VALUE='$resources.GoBackToOrderPageButton$'>

           $data.hiddenFormShowType$
           $data.hiddenFormSortType$
           $data.hiddenFormCommand$
        </FORM>
    </TD>

   </TR>
</TABLE>
</CENTER>


<P>&nbsp;
<BR>&nbsp;
```

```
</BODY>
</HTML>

*/

}


addRemoveComments(history, command) {

  Sub resources, data

  if ( history == "" ) {

      /*


     */
   }
  else

   if ( command == "merchant" )
    {
      /*

      <TABLE BORDER=0  COLS=1 WIDTH="602" BGCOLOR="#6699CC" >
        <TR>
           <TD><B>$resources. CommentInfo$</B></TD>
        </TR>
      </TABLE>
       <BR>
       <FORM NAME='comment'>
          <TEXTAREA  rows=10 cols=50 WRAP=on>
            $history$
          </TEXTAREA>
       </FORM>

       <FORM NAME='removeComment' METHOD='POST' ACTION='servlet/MerchantAdmin'
onSubmit='return reallyRemove()'>

         <!-- Hidden field to make sure the browser does not cache the confirm page -->
         <SCRIPT>
             var now = new Date();
            document.writeln('<INPUT TYPE=hidden NAME="CTS"  VALUE= "' + now.getTime() +
'">');
         </SCRIPT>

         <INPUT TYPE='hidden' NAME='GOTO' VALUE='InvoicePage'>
         <INPUT TYPE='submit' NAME='removeBtn' VALUE='$resources.RemoveComments$'>
        <INPUT TYPE='hidden' NAME='SelectedOrders' VALUE='$data.orderNumber$'>
         <INPUT TYPE='hidden' NAME='SelectedType'   VALUE='$data.SelectedType$'>

         $data.hiddenFormShowType$
         $data.hiddenFormSortType$
         $data.hiddenFormCommand$
       </FORM>

     */
    }
  else {

    /*
       <B>$resources.CommentInfo$</B>
       <BR>
       <FORM NAME='comment'>
          <TEXTAREA  rows=10 cols=50 WRAP=on>
            $history$
          </TEXTAREA>
       </FORM>

    */
  }

}


-- inserted for Gift Message customization

addGiftMessage(giftmsg) {


  if ( giftmsg != "null" ) {

      /*

      <TABLE BORDER=0  COLS=1 WIDTH="602" BGCOLOR="#6699CC" >
        <TR>
           <TD><B>Gift message</B></TD>
        </TR>
      </TABLE>
       <BR>
       <FORM NAME='giftmessage'>
          <TEXTAREA  rows=10 cols=50 WRAP=on>
$giftmsg$
          </TEXTAREA>
       </FORM>


    */
  }

}


addChangeStatusButton(type, command, stype)
{
  Sub resources, data

  if ( type == "Cancel" || type == "Shipped" || type == "Cancelled" ) {
        /*

        */
    }
  else
    if ( command == "merchant" )
     {

        /*
           <FORM NAME='changeStatus' METHOD='POST' ACTION='servlet/MerchantAdmin'
onsubmit='getOrderType("$stype$")'>

           <!-- Hidden field to make sure the browser does not cache the confirm page -->
           <SCRIPT>
              var now = new Date();
              document.writeln('<INPUT TYPE=hidden NAME="CTS"  VALUE= "' + now.getTime()
+ '">');
```

```
           </SCRIPT>


           <INPUT TYPE='hidden' NAME='GOTO' VALUE='ChangeStatus'>
           <INPUT TYPE='submit' NAME='changeStatus'
VALUE='$resources.ChangeStatusButton$'>
       <INPUT TYPE='hidden' NAME='SelectedOrders' VALUE='$data.orderNumber$'>
           <INPUT TYPE='hidden' NAME='SelectedType'   VALUE=''>
           $data.hiddenFormShowType$
           $data.hiddenFormSortType$
           $data.hiddenFormCommand$

           </FORM>

         */
    }
   else
     if ( command == "shopper" && type ==  "Pre-authorized" )
       {

          /*

           <FORM NAME='changeStatus' METHOD='POST' ACTION='servlet/MerchantAdmin'
onsubmit='getOrderType("$stype$)'>

           <!-- Hidden field to make sure the browser does not cache the confirm page -->
           <SCRIPT>
              var now = new Date();
              document.writeln('<INPUT TYPE=hidden NAME="CTS"  VALUE= "' + now.getTime()
+ '">');
           </SCRIPT>


           <INPUT TYPE='hidden' NAME='GOTO' VALUE='ChangeStatus'>
           <INPUT TYPE='submit' NAME='changeStatus'
VALUE='$resources.ChangeStatusButton$'>
       <INPUT TYPE='hidden' NAME='SelectedOrders' VALUE='$data.orderNumber$'>
           <INPUT TYPE='hidden' NAME='SelectedType'   VALUE=''>
           $data.hiddenFormShowType$
           $data.hiddenFormSortType$
           $data.hiddenFormCommand$

           </FORM>

         */

      }
    else {
         /*

         */
     }
}
```

Appendix F. A MultiPurpose Code Generation language

F.1 Purpose

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.

F.2 Introduction

F.2.1 What is MPG?

MPG (MultiPurpose Generator) 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 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, etc. The standard letter, in this case, would be the template and the model would consist of the 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 (Note this is the same as /* Enterprise Software */ date /* .... */). Variables are pulled in from the model by declaring them as: Model var1, var2 etc.

F.2.2 Why MPG?

This is a question that I have often been asked and have often asked myself in the past. This question 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 like ..." + 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). Handling lists in general proved awkward. After time, (with many developers editing the same code) the model/view separation in the code started to deteriorate.

It was these any 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 lifecycle
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)

F.3 Data Model

In this section, we will describe how to create the model that is to be used in the template.

F.3.1 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.

F.3.2 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();
```

F.3.2.1 Using regular Java objects

Regular Java objects (ie 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* etc 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));
```

F.3.2.2 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 sub-structures.

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);
```

F.3.2.3 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.

F.3.2.4 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()*.

F.3.3 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" ]
```

F.4 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 get up and running quickly using MPG.

F.4.1 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.

F.4.1.1 Case Sensitivity

MPG is a case-sensitive language. Keywords, variable names and procedure names must be typed with consistent captalization. For example, the variable x is entirely different from the variable X.

F.4.1.2 Whitespace

The MPG parser ignores spaces, tabs and newlines that appear between tokens in the templates. Note that this does not include whitespace that included in strings (delimited by ‘ and “) and whitespace that exists inside the freeform delimiters (/* and */).

F.4.1.3 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 freeform delimiters.

F.4.1.4 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. Semi-colons may be added to statements, but will be ignored.

F.4.1.5 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 newline to a string literal, use \n ('line1\nline2'). Boolean values *true* and *false* may also appear as literals in the template.

F.4.1.6 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 are 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 tilda 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.

F.4.1.7 Identifiers

An identifier is simply a name. In MPG, identifiers are used to name variables and functions. The rules for legal identifiers are the similar to that 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 ord Latin-1 characters are not allowed.

F.4.1.8 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	

F.4.2 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 sub-scopes. 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
```

F.4.2.1 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.

F.4.3 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).

F.4.4 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 that the variable refers to. 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 (apply to a single expression) and binary operators (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.

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, substraction
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
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)

F.4.5 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 interal cursor on a list). To add logical flow to a template you need to use statements.

F.4.5.1 4.5.1 *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) {

}
...
```

F.4.5.2 *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 and 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 (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 below) 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 on 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 to occur. 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, then 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
}
```

F.4.5.3 *while* loop

While loops may be used to repeat a segment of code while a certain condition remains true. This is used the same as the while loop in C.

Example:

```
while(  i > 0 ) {
   i = i-1
}
```

F.4.5.4 *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.

F.4.5.5 *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.

F.4.5.6 *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.

F.4.6 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) that the procedure is to operate upon. 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 (ie 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).

F.4.7 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 by 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)
```

The following list describes all of the 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
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 occurences of the *pattern* with the replacement string *str*
change(pattern, str, max)	Replaces up to *max* occurences of the *pattern* with the replacement string *str.*
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 occurences of the specified pattern
indexOf(pattern)	Returns the index of the given pattern (index starts at 0)
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)
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
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 javascript 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

F.4.8 4.10 Including other templates

TBD

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. Sub-directories 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 G. Customization of NCHS on Windows NT

G.1 Customizing store creation process on Windows NT

The following steps will guide you in adding this customization to Net.Commerce Hosting Server:

1. Make a copy of the **Store Information** form to be inserted into the store creation flow.

   The **Store Information** form layout is contained in the StoreInfo.tem file. This file defines how the form looks, what text is used and the actions the form performs. A copy of this file will be modified and inserted into the store creation flow.

   Make a copy of the StoreInfo.tem file located in the `<drive>`:\ibm\NetCommerce3\Tools\mpg_templates\nchs\mtool\ directory, (where `<drive>` is the letter of the drive where Net.Commerce is installed). Name this copy Register2.tem.

Note:
Make sure that the file being modified in step 2 is Register.tem and the file being modified in step 3 is Register2.tem.

2. Modify the **Store Creation** form to present the new **Store Information** form after it is submitted.

   The **Store Creation** form layout is contained in the Register.tem file. This file defines how the form looks, what text is used and the actions the form performs. This file will be modified to call the new **Store Information** form instead of the store creation command.

   Open the file Register.tem in a text editor and comment out the following line, by adding -- to the beginning of each line. This will disable the call to the store creation command by this form.

```
-- /*
--<SCRIPT>top.location.href='http://$env.hostname$/servlet/MerchantAdmi
n?PROCESS=CTnchs.mtool.Filter&XMLFile=nchs.mtool.merchantTool.xml&start
ingFolder=getStartedFolder'; </SCRIPT>
```

```
-- */
```

To direct this form to the new **Store Information** form, add the following line just above the commented out line:

```
/*
<SCRIPT>location.href='http://$env.hostname$/servlet/MerchantAdmin?DISP
LAY=CTnchs.mtool.Register2'; </SCRIPT>
*/
```

Save this file and exit.

3. Modify the new **Store Information** form to call the store creation command.

   Open the file Register2.tem in a text editor and comment out the following line, by adding -- to the beginning of each line. This will disable the displaying of the merchant tool **Get Started** tab.

```
-- /*
--<SCRIPT>window.location="http://$env.hostname$/servlet/MerchantAdmin?
DISPLAY=CTnchs.mtool.StoreInfoConfirm"; </SCRIPT>
-- */
```

   In order to direct the new **Store Information** form to call the store creation command, add the following line after the commented out line:

```
/*
<SCRIPT>top.location.href='http://$env.hostname$/servlet/MerchantAdmin?
PROCESS=CTnchs.mtool.Filter&XMLFile=nchs.mtool.merchantTool.xml&startin
gFolder=getStartedFolder'; </SCRIPT>
*/
```

   To display the correct error messages during the new step in the store creation process, find the line containing the `$mtoolNLS.storeInfoErrorMandatoryTop$` variable and change it to `$mtoolNLS.errorMandatoryTop$`, then find the line containing the `$mtoolNLS.storeInfoErrorMandatoryEnd$` variable and change it to `$mtoolNLS.errorMandatoryEnd$`.

   In order to disable the call to the old task, comment out the following line by adding -- to the beginning of the line:

```
-- <INPUT TYPE=hidden NAME="PROCESS" VALUE="CTnchs.mtool.StoreInfo">
```

Now add a similar line beneath the commented out line to call the new Register2 task. This tells the form which task to use for processing.

```
<INPUT TYPE=hidden NAME="PROCESS" VALUE="CTnchs.mtool.Register2">
```

Save this file and exit.

4. Add the new store creation step to the mtoolTasks.xml file so that it will be recognized by Net.Commerce Hosting Server.

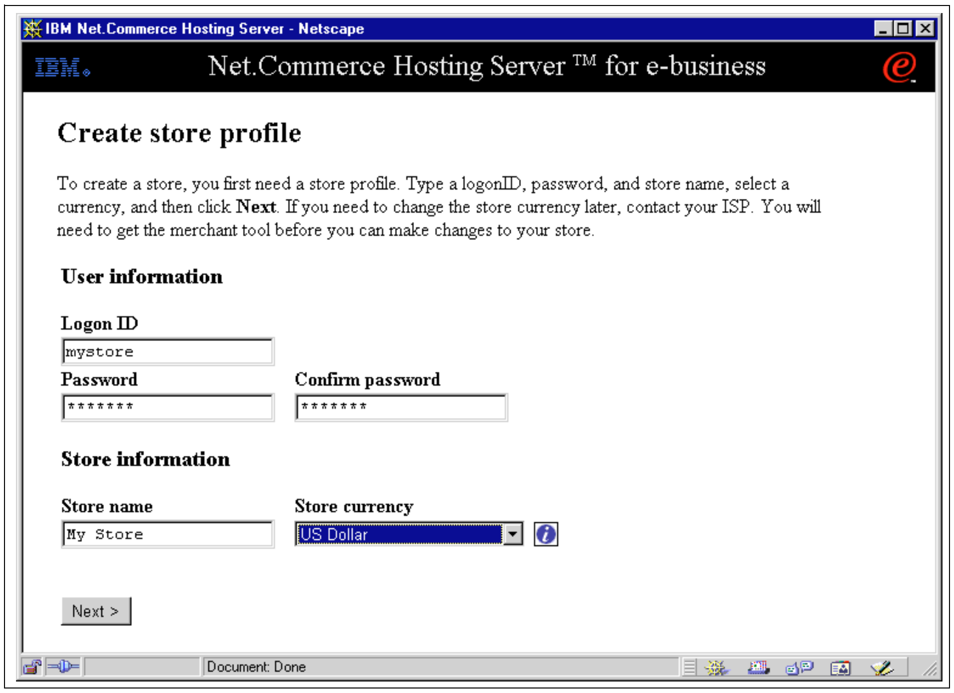
   The mtoolTasks.xml file contains a listing of the xml tasks that Net.Commerce Hosting Server recognizes. This file contains the necessary information about each task such as file locations, required parameters and access controls.

   Open the mtoolTasks.xml file in the `<drive>`:\ibm\NetCommerce3\Tools\config\nchs\ directory, (where `<drive>` is the letter of the drive where Net.Commerce is installed), and add the following lines at the bottom of the file, just above the `</taskConfig>` line. This will register the new step in the store creation process with Net.Commerce Hosting Server and direct it to the proper files.

   ```
   <task name="CTnchs.mtool.Register2"
   template="nchs/mtool/Register2.tem"
   dbSessionRequired="true"
   requiredProcessParams="contactEMail1"/>
   ```

   If the CSP wants to require other fields in addition to the e-mail field, they can be added to the `requiredProcessParams` list. Save this file and exit.

5. Stop and restart the Net.Commerce instance, administrator server and webserver as directed in *"Installing and Getting Started Guide",* GC09-2808-01. Figure 42 on page 184 and Figure 43 on page 185 show the new store creation flow.


IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

Create store profile

To create a store, you first need a store profile. Type a logonID, password, and store name, select a currency, and then click **Next**. If you need to change the store currency later, contact your ISP. You will need to get the merchant tool before you can make changes to your store.

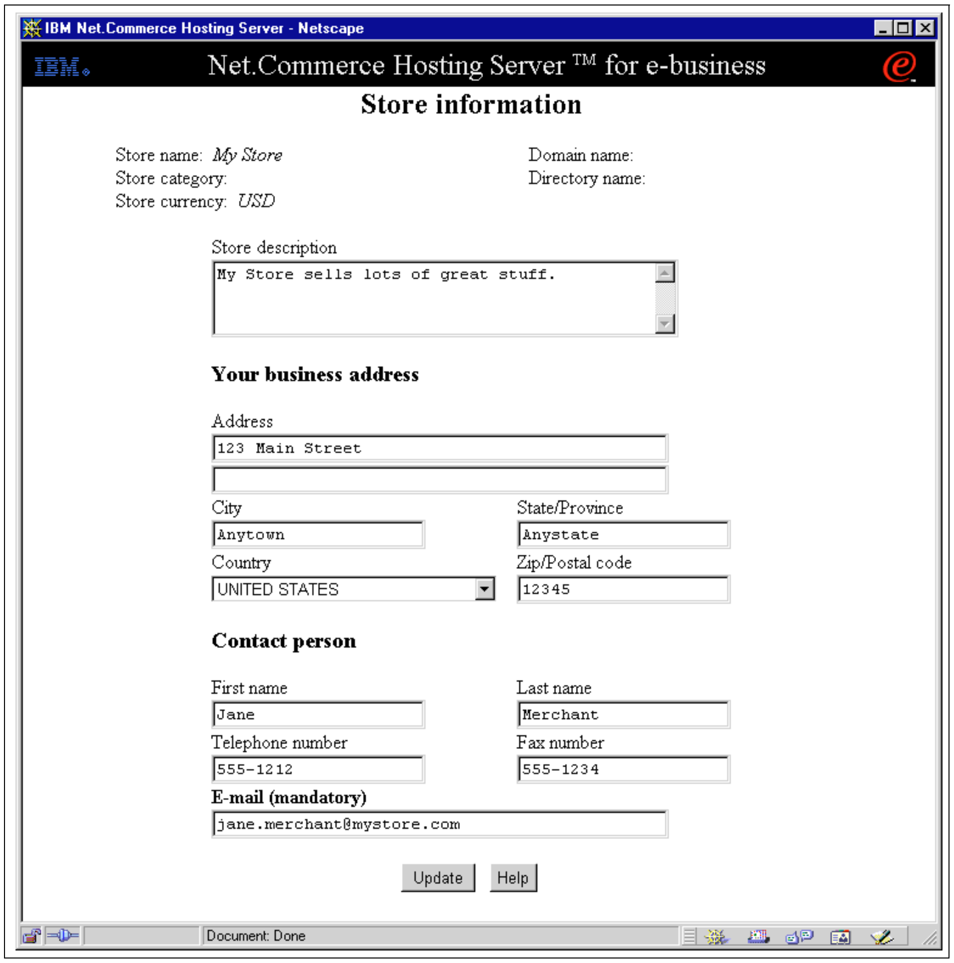
User information

Logon ID	mystore
Password	*******
Confirm password	*******

Store information

Store name	My Store
Store currency	US Dollar

Figure 42. Store creation form.


IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

Store information

Store name: *My Store*
Store category:
Store currency: *USD*

Domain name:
Directory name:

Store description
My Store sells lots of great stuff.

Your business address

Address	123 Main Street
City	Anytown
State/Province	Anystate
Country	UNITED STATES
Zip/Postal code	12345

Contact person

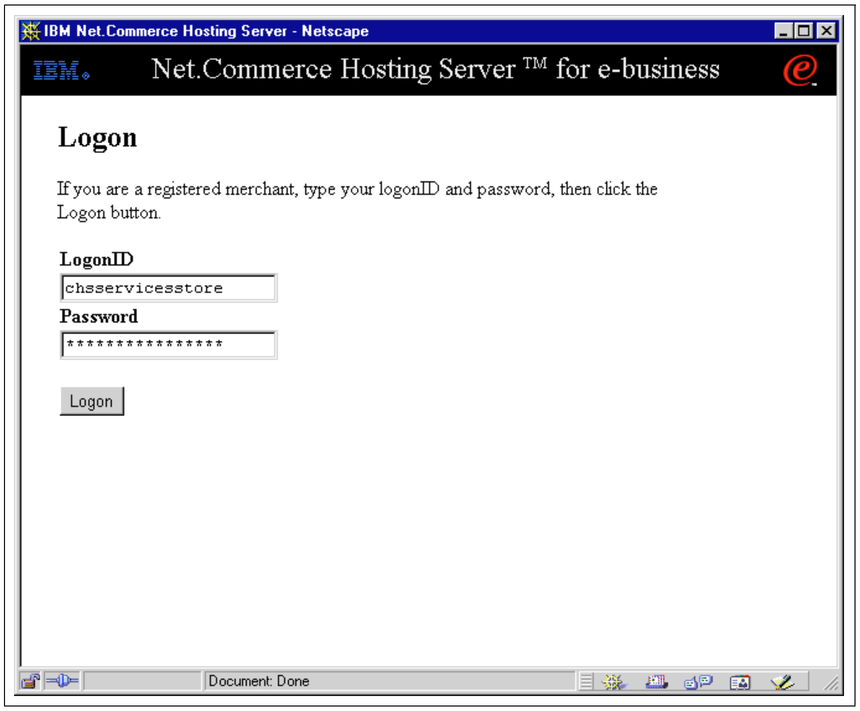
First name	Jane
Last name	Merchant
Telephone number	555-1212
Fax number	555-1234
E-mail (mandatory)	jane.merchant@mystore.com

Figure 43. Newly added Store Information form.

G.2 Restricting creation of merchant store on Windows NT

1. Make store creation access available for purchase from your Services Store site by adding a store creation access product to your catalog.

   Open the cspsite and click on **Manage Store**. Log on as the Commerce Hosting Server, (CHS), services store manager, (default logon and password is `chsservicesstore`). as shown in Figure 44 on page 186.



IBM Net.Commerce Hosting Server - Netscape

Net.Commerce Hosting Server ™ for e-business

Logon

If you are a registered merchant, type your logonID and password, then click the Logon button.

LogonID

chsservicesstore

Password

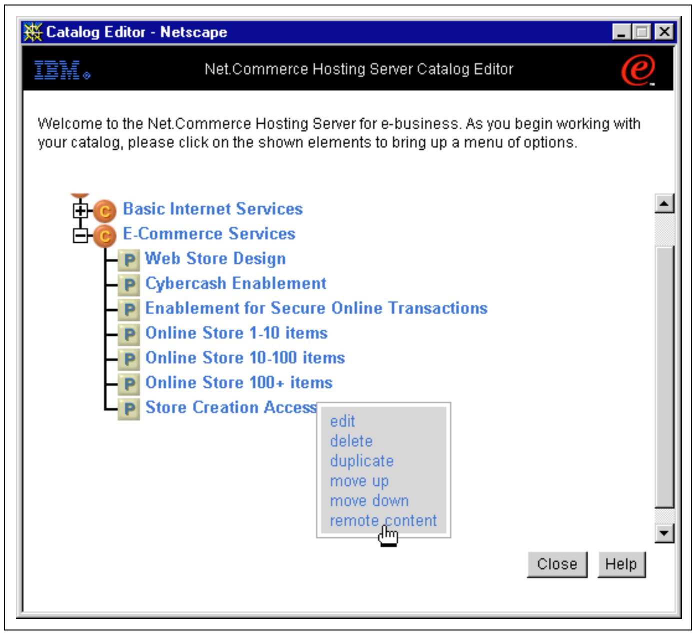
****

Logon

Document: Done

Figure 44. CHS services store logon screen.

Add an item to the catalog for store creation access by clicking **Edit Catalog**. Name the new item "Store Creation Access". Before exiting the **Catalog Editor**, be sure to write down the url **Add to Shopping Cart link**. It will be needed in the next step. To obtain this url, click once on the newly created Store Creation Access product and click **Remote Content**, as shown in Figure 45 on page 187. Copy the url under **Add to Shopping Cart link**.


Catalog Editor - Netscape

Net.Commerce Hosting Server Catalog Editor

Welcome to the Net.Commerce Hosting Server for e-business. As you begin working with your catalog, please click on the shown elements to bring up a menu of options.

*Figure 45. Obtaining the **Add to Shopping Cart link**.*

In addition to needing this url for the next step, there are two pieces of information that you will need in later steps that should be obtained at this time. In the url, there are name/value pairs. Extract the store reference number and the product SKU number from these name/value pairs. For example:

```
http://<hostname>/servlet/ShoppingCart?merchant.refno=XXX&product.SKU=YYY
```

In this example, **XXX**, and **YYY** are the store reference number and product SKU number, respectively.

Exit the **Catalog Editor** and then publish the CHS Services Store by clicking on **Publish Store**.

2. Change the logic of the store creation process so that a check for access privilege is made prior to allowing access to store creation.

   When the prospective merchant clicks **Create Store** the objective is to only allow access to store creation if it has been purchased. The **Create Store** link calls a static html file will displays the **Create Store** form. To disable store creation access until access has been purchased, this link will be changed to call macro instead. This macro will check to see if store creation access has been purchased and will then display the appropriate screen.

To create a macro that checks for store creation privileges, open a text editor and type the following, (adding the **Add to Shopping Cart link** where instructed):

```
%{==================================================================
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 fictitious and
any similarity to the names and addresses used by actual persons or
business enterprises is entirely coincidental.

Licensed Materials - Property of IBM

5697-D24

(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

==================================================================%}

%function(dtw_odbc) check_status() {
SELECT shfield2, shlogid
FROMshopper
WHERE shlogid='$(SESSION_ID)'

%REPORT {
%ROW {
 @DTW_assign(STATUSFIELD, V_shfield2)
%}
%}
```

```
%}

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

%HTML_REPORT{
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
<meta NAME="keywords"
CONTENT="Net.Commerce, Commerce Hosting Server, internet, internet
service provider, web hosting, dial up access">
<title>IBM Net.Commerce Hosting Server</title>
</head>
<body>
@check_status()

%if (STATUSFIELD != "1")
<TABLE>
<TR>
<TD>To create a store, click the link below and purchase access for only
$25!</TD>
</TR>
<TR>
<TD>Please click <A href="*** Enter Add to Shopping Cart url here
***">here</A>
to purchase store creation access.</TD>
</TR>
</TABLE>
%else
<script Language="JavaScript">

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.";

function create_store() {
```

```
if ((navigator.appName.indexOf("Netscape")  > -1 &&
parseFloat(navigator.appVersion) >= 4.06) ||
(navigator.appName.indexOf("Microsoft") > -1 &&
parseFloat(navigator.appVersion) >= 4.0)) {
document.write("<p>" + launch_str+ "<br>" );

var url = "/servlet/MerchantAdmin?";
var target = window.open("", "MerchantTool",
"resizable=yes,scrollbars=yes,status=yes,width=750,height=500,screen=0,
screenY=0,left=0,top=0");

if (target.document.URL.indexOf(url) == -1)
target.location.href = url + "GOTO=Banner&body=RegisterPage";
target.focus();
} else {
document.write("<p>" + unsupported_browser_text1);
document.write("<ul>");
document.write("<li>" + unsupported_browser_item1 + "</li>");
document.write("<li>" + unsupported_browser_item2 + "</li>");
document.write("</ul>");
document.write(unsupported_browser_text2);
}
}
create_store();
</script>
%endif

<br>
<br>
<br>
<br>
<font SIZE="1" FACE="Verdana, Arial, Helvetica">Please send all
inquiries to:&nbsp;<a
HREF="mailto:webmaster@CHSNet.com">webmaster@CHSNet.com</a><br>
Site comments:&nbsp;<a
HREF="mailto:webmaster@CHSNet.com">webmaster@CHSNet.com</a><br>
© 1998, IBM Net.Commerce for CHS.<br>
</font></p>
</body>
</html>
%}
```

Save this file as creation_access.d2w in the

<drive>:\ibm\NetCommerce3\macro\en_US\ncadmin\sitemgr\ directory, (where <drive> is the letter of the drive where Net.Commerce is installed).

3. Edit the navigation.html file to change the link for **Create Store**.

   The **Create Store** link in the navigation.html file calls a static html file will displays the **Create Store** form. To disable store creation access until access has been purchased, this link will be changed to call the newly created creation_access.d2w macro instead.

   Use a text editor to open the file navigation.html in the <drive>:\ibm\NetCommerce3\html\en_US\cspsite\ directory, (where <drive> is the letter of the drive where Net.Commerce is installed).

   In order to change the **Create Store** link, comment out the link to the create_store.html file by placing HTML comment tags around this section:

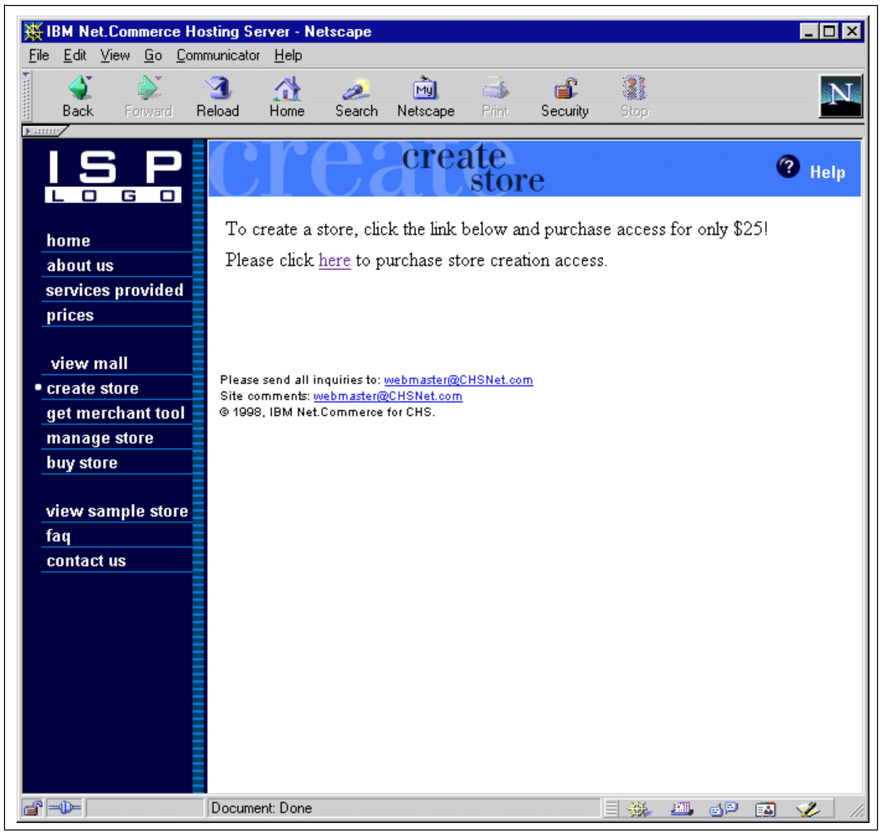
```
<!-- create store -->
<!-- <TR>
<TD><IMG NAME="item6" SRC="/CHS/images/bullet_blank.gif" WIDTH=5
HEIGHT=5 ALT="" BORDER=0></TD>
<TD NOWRAP>&nbsp;<B><A
HREF="javascript:go(6,'/cspsite/create_store_banner.html','/cspsite/cre
ate_store.html')" onMouseOver="on(6); status='create store'; return
true;" onMouseOut="off(6); status='';">create store</A></B></TD>
</TR>
<TR>
<TD></TD>
<TD><IMG SRC="/CHS/images/separator.gif" WIDTH=122 HEIGHT=1 ALT=""
BORDER=0></TD>
</TR> -->
```

   Replace this commented out section with a new, similar section containing a link to the creation_access.d2w macro:

```
<TR>
<TD><IMG NAME="item6" SRC="/CHS/images/bullet_blank.gif" WIDTH=5
HEIGHT=5 ALT="" BORDER=0></TD>
<TD NOWRAP>&nbsp;<B><A HREF="javascript:go(6,
'/cspsite/create_store_banner.html',
'/cgi-bin/ncommerce3/ExecMacro/ncadmin/sitemgr/creation_access.d2w/repo
rt?merfnb=$env.merchant_id$')" onMouseOver="on(6); status='create
store'; return true;" onMouseOut="off(6); status='';">create
```

```
store</A></B></TD>
</TR>
<TR>
<TD></TD>
<TD><IMG SRC="/CHS/images/separator.gif" WIDTH=122 HEIGHT=1 ALT=""
BORDER=0></TD>
</TR>
```

Save the navigation.html file and exit. Figure 46 shows the new screen that will appear when a merchant clicks **Create Store**.



To create a store, click the link below and purchase access for only $25!

Please click here to purchase store creation access.

Please send all inquiries to: webmaster@CHSNet.com
Site comments: webmaster@CHSNet.com
© 1998, IBM Net.Commerce for CHS.

Figure 46. New store creation access screen.

4. Modify the ord_ok.d2w macro to enable store creation after access has been purchased.

   The ord_ok.d2w macro displays the order confirmation screen. When store creation access has been purchased, the shfield2 field in the shopper table should be set to 1 to indicate that the prospective merchant has purchased access and can therefore be granted access to store creation.

This update can be performed by checking the list of products purchased in the product list and if any of those products is the store creation access product, the shfield2 field is set to 1 indicating access has been purchased. After this update to the database has been performed, a link to the store creation command will be displayed so that the merchant can then create their store. This link is only available when store creation access has been purchased and is only active for that particular session. This prevents a registered merchant from returning to the cspsite and creating additional stores.

To add this functionality to the ord_ok.d2w macro, open this file which is located in the `<drive>`:\ibm\NetCommerce3\macro\en_US\\`<ref num>`\ directory, (where `<drive>` is the drive on which Net.Commerce is installed and `<ref num>` is the reference number of the CHS Services Store obtained in Step 1).

In this file, make the following changes which appear in **boldface**, (replace all `<ref num>` occurrences with the CHS Services Store reference number).

```
%include "<ref num>\include.inc"
%include "<ref num>\new_ord_ok.d2w"
```

Save this file and exit. The changes made to the ord_ok.d2w macro instruct Net.Commerce to include a new macro called new_ord_ok.d2w. This macro will be very similar to the existing ord_ok.d2w macro in the \ibm\NetCommerce3\macro\common\CSPstoremodel\ directory except for the addition of the logic to update the database with a flag to grant store creation access.

In order to create this new macro, make a copy of the ord_ok.d2w macro in the `<drive>`:\ibm\NetCommerce3\macro\common\CSPstoremodel\ directory, (where `<drive>` is the drive on which Net.Commerce is installed), and name this copy new_ord_ok.d2w. Move this new copy to the `<drive>`:\ibm\NetCommerce3\macro\en_US\\`<ref num>`\ directory, (where `<drive>` is the drive on which Net.Commerce is installed and `<ref num>` is the reference number of the CHS Services Store). Open the new_ord_ok.d2w macro file in a text editor and make the following additions/changes which appear in **boldface**. Insert the product SKU number obtained in Step 1 where instructed.

```
%{=================================================================
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 fictitious and
any similarity to the names and addresses used by actual persons or
business enterprises is entirely coincidental.

Licensed Materials - Property of IBM 5697-D24 (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
==================================================================%}
%INCLUDE "/CSPstoremodel/translation_text.inc"
%INCLUDE "/CSPstoremodel/format_pricedefinition.inc"
%INCLUDE "$(DirectoryName)/navbar.inc"

%define {
SHOWSQL="NO"
CreationFlag="False"
%}

%INCLUDE "ord_taxshiprules.inc"

%function(dtw_odbc) UPDATE_CREATION_ACCESS(){
UPDATE shopper
SET shfield2 = '1'
WHERE shlogid = '$(SESSION_ID)'
%}

%function(dtw_odbc) GET_ORBILLTO (){
SELECT orbllto
FROM orders
WHERE orrfnbr = $(order_rn)

%REPORT {
%ROW {
```

```
@dtw_assign(BILLING_ADDRESS_RN, V_orbllto)
%}
%}
%MESSAGE{
100: { %} :CONTINUE
default: { ERROR in GET_ORBLLTO %}
%}
%}

%function(dtw_odbc) IS_SET ()
{
SELECT ompaymthd, setsstatcode, setsfailtype
FROM ordpaymthd, setstatus, orders
WHERE omornbr = $(order_rn) and setsornbr = $(order_rn) and orrfnbr =
setsornbr and orshnbr = $(SESSION_RN)

%REPORT {
<CENTER>
<TABLE width=530 CELLPADDING=4 CELLSPACING=0 BORDER=0 ALIGN="center">
<TR>
<TD ALIGN="left" VALIGN="center">
%ROW {
@dtw_assign(BILLING_ADDRESS_RN, V_orbllto)
@dtw_assign(PAYMENT_METHOD, V_ompaymthd )
<B>$(TXT_THANKYOU)</B>
<BR>
%INCLUDE "ord_set_returncodes.inc"
%}
</TD>
</TR>
</TABLE>
%}
%MESSAGE{
100: { %} :CONTINUE
default: { ERROR in IS_SET %}
%}
%}

%function(dtw_odbc) GET_SHOPPER_TYPE() {
select shshtyp
from shopper
where shrfnbr = $(SESSION_RN)
```

```
%REPORT{
%ROW{
 @DTW_assign(SHOPPER_TYPE, V_shshtyp)
%}
%}
%MESSAGE{
default: { ERROR in GET_SHOPPER_TYPE %}
%}
%}

%function(dtw_odbc) SHOPPER_INFO() {
select sarfnbr, salname, safname, saaddr1, saaddr2, sacity, sastate,
sazipc, sacntry
from shaddr
where sashnbr=$(SESSION_RN) and sarfnbr=$(BILLING_ADDRESS_RN)

%REPORT{
<CENTER>
<TABLE width=530 CELLSPACING=0 CELLPADDING=4 BORDER=0 ALIGN="center">
%ROW{
%IF (($(PAYMENT_METHOD) != "SET") && ($(PAYMENT_METHOD) != "SETNV") )
<TR>
<TD COLSPAN=3>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
<B>$(TXT_THANKYOU)</B>
   </FONT>
   </TD>
   </TR>
   <TR>
<TD COLSPAN=3>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CONF_MSG)</FONT>
</TD>
</TR>
%ENDIF
<TR><TD><BR><BR></TD></TR>
<TR>
    <TD COLSPAN=3 ALIGN="center" bgcolor="#E0E0E0">
<B>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
$(LBL_ORDERNUMBER) : $(order_rn)
%IF (SHOPPER_TYPE == "G")Z--
$(LBL_CUSTOMERCODE) :  $(SESSION_ID)
%ENDIF
</FONT>
</B>
</TD>
</TR>
<TR><TD><BR></TD></TR>
   <TR>
<TD>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(TXT_MAILTO)
 </B></FONT>
</TD>
<TD width=10></TD>
<TD>
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(TXT_MAILSHIPTO)</B></FO
 NT>
</TD>
</TR>
   <TR>
<TD BGCOLOR=white width=260 VALIGN=top>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
%INCLUDE "/CSPstoremodel/address_static.inc"
</FONT>
</TD>
<TD width=10></TD>
 %}
 %}
 %MESSAGE{default: { ERROR in SHOPPER_INFO %} :CONTINUE
 %}
 %}

 %function(dtw_odbc) SHOPPER_SHIPTO_INFO() {
 select salname, safname, saaddr1, saaddr2, sacity, sastate, sazipc,
 sacntry
 from shaddr, shipto
 where stornbr=$(order_rn) and stsanbr=sarfnbr

 %REPORT{
<TD BGCOLOR=white  width=260 VALIGN=top>
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
%INCLUDE "/CSPstoremodel/address_static.inc"
</FONT>
</TD>
</TR>
```

```
</TABLE>
</CENTER>
%}

%MESSAGE{default: { ERROR in SHOPPER_SHIPTO_INFO %}
%}
%}

%function(dtw_odbc) DISPLAY_DETAILS_LIST() {
select strfnbr, stsanbr, stshnbr, stmenbr, stprnbr, stprice, stquant,
stcpcur, prrfnbr, prnbr, prldesc2, prsdesc, salname, safname
from shipto, product, shaddr
where stshnbr=$(SESSION_RN) and stmenbr=$(MerchantRefNum) and
stprnbr=prrfnbr and stornbr=$(order_rn)
and stsanbr=sarfnbr
order by stmenbr, stsanbr, strfnbr

%REPORT{
<BR>
<CENTER>
<TABLE width=530 CELLPADDING=4 CELLSPACING=0 BORDER=0 ALIGN="center">
<TR>
    <TD ALIGN=left VALIGN=top><B><FONT
SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_PRODUCTNUM)</FONT></B>
</TD>
<TD ALIGN=left VALIGN=top><B><FONT
SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_PRODUCTNAME)</FONT></B
></TD>
<TD ALIGN=middle VALIGN=top><B><FONT
SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_QUANTITY)</FONT></B></
TD>
    <TD ALIGN=right VALIGN=top><B><FONT
SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_PRODUCTPRICE)
%IF (CurDescription != null)
<BR>[$(CurDescription)]
%ENDIF
</FONT></B></TD>
<TD ALIGN=right VALIGN=top><B><FONT
SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(LBL_SUBTOTAL)
%IF (CurDescription != null)
<BR>[$(CurDescription)]
%ENDIF
</FONT></B></TD>
```

```
</TR>
<TR><TD colspan=5><HR></TD></TR>
 %ROW{
<TR>
@DTW_FORMAT(V_stprice, "", CurDecimalPlaces, FORMATTEDPRODPRICE)
@DTW_MULTIPLY(V_stquant, V_stprice, SUB_TOT)
@DTW_FORMAT(SUB_TOT, "", CurDecimalPlaces, FORMATTEDSUBTOTPRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDSUBTOTPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSUBTOTPRICE,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDPRODPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDPRODPRICE,OUT_PRICE)

%if ($(V_prnbr) == "**insert product SKU number here**")
@DTW_ASSIGN(CreationFlag, "True")
%endif

    <TD ALIGN=left><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(V_prldesc2) - </FONT></TD>
<TD ALIGN=left><FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
 $(V_prsdesc)</FONT></TD>
<TD ALIGN=middle><FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
 $(V_stquant)</FONT></TD>
    <TD ALIGN=right><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDPRODP
 RICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_stcpcur)
%ENDIF
</FONT></TD>
    <TD ALIGN=right ALIGN="right"><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(CurPrefix)$(FORMATTEDSU
 BTOTPRICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_stcpcur)
%ENDIF
</B></FONT></TD>
</TR>
<TR><TD HEIGHT=5></TD></TR>
 %}
 <TR><TD colspan=5><HR></TD></TR>
 %}
```

```
%MESSAGE{
100    : {<BR><FONT
SIZE=3><B>$(MSG_ORDERLIST_EMPTY)</B></FONT>%}:continue
default: {ERROR : Problem with DISPLAY_DETAILS_LIST function %}
%}
%}

%function(dtw_odbc) DISPLAY_CHARGES_MerchantTax() {
select distinct orprtot, ortxtot, orshtot, orshtxtot, orcpcur,
(oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as mttax
from orders, orderpay, shaddr
whereormenbr=$(MerchantRefNum) and oysanbr=sarfnbr and
orrfnbr=$(order_rn) and oyornbr=$(order_rn)
%REPORT{
%ROW{
@DTW_FORMAT(V_orprtot, "", CurDecimalPlaces, FORMATTEDSUBTOTPRICE)
@DTW_FORMAT(V_ortxtot, "", CurDecimalPlaces, FORMATTEDTAXTOT)
@DTW_FORMAT(V_orshtot, "", CurDecimalPlaces, FORMATTEDSHIPTOT)
@DTW_FORMAT(V_orshtxtot, "", CurDecimalPlaces, FORMATTEDSHIPTAXTOT)
@DTW_ADD(V_orprtot, V_ortxtot, total)
@DTW_ADD(total, V_orshtot, total)
@DTW_ADD(total, V_orshtxtot, total)
@DTW_FORMAT(total, "", CurDecimalPlaces, FORMATTEDTOTPRICE)
%IF (ConvMultOrDiv == "M")
@DTW_MULTIPLY(FORMATTEDTOTPRICE, ConvFactor, CONVPRICE)
@DTW_FORMAT(CONVPRICE, "", ConvCurDecimalPlaces, CONVFORMATTEDPRICE)
%ELIF (ConvMultOrDiv == "D")
@DTW_DIVIDE(FORMATTEDTOTPRICE, ConvFactor, CONVPRICE)
@DTW_FORMAT(CONVPRICE, "", ConvCurDecimalPlaces, CONVFORMATTEDPRICE)
%ENDIF
@DTW_ASSIGN(IN_PRICE,FORMATTEDSUBTOTPRICE)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSUBTOTPRICE,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDTAXTOT)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDTAXTOT,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDSHIPTOT)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSHIPTOT,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDSHIPTAXTOT)
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDSHIPTAXTOT,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,FORMATTEDTOTPRICE)
```

```
%INCLUDE "/CSPstoremodel/format_price.inc"
@DTW_ASSIGN(FORMATTEDTOTPRICE,OUT_PRICE)
@DTW_ASSIGN(IN_PRICE,CONVFORMATTEDPRICE)
%INCLUDE "/CSPstoremodel/format_convprice.inc"
@DTW_ASSIGN(CONVFORMATTEDPRICE,OUT_PRICE)
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_SUBTOTAL)
 </B></FONT></TD>
    <TD ALIGN="right"><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSUBTO
 TPRICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_orcpcur)
%ENDIF
</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_TAX)
 </B></FONT></TD>
    <TD ALIGN="right">
%IF (TAXRULE_EXISTS == "YES" && CurDescription == null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDTAXTO
 T)$(CurPostfix) $(V_orcpcur)</FONT>
%ELIF (TAXRULE_EXISTS == "YES" && CurDescription != null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDTAXTO
 T)$(CurPostfix)</FONT>
%ELSE
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>-----</FONT>
%ENDIF
</TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_SHIPPING)
 </b></FONT></TD>
    <TD ALIGN="right">
%IF (SHIPRULE_EXISTS == "YES" && CurDescription == null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 OT)$(CurPostfix) $(V_orcpcur)</FONT>
%ELIF (SHIPRULE_EXISTS == "YES" && CurDescription != null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 OT)$(CurPostfix) </FONT>
%ELSE
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>-----</FONT>
%ENDIF
</TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_SHIPPINGTAX)
 </b></FONT></TD>
    <TD ALIGN="right">
%IF (SHIPRULE_EXISTS == "YES" && CurDescription == null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 AXTOT)$(CurPostfix) $(V_orcpcur)</FONT>
%ELIF (SHIPRULE_EXISTS == "YES" && CurDescription != null)
<FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>$(CurPrefix)$(FORMATTEDSHIPT
 AXTOT)$(CurPostfix)</FONT>
%ELSE
<FONT SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>-----</FONT>
%ENDIF
</TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(LBL_TOTAL)
 </B></FONT></TD>
    <TD ALIGN="right" BGCOLOR="white"><FONT
 SIZE=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)><B>$(CurPrefix)$(FORMATTEDTO
 TPRICE)$(CurPostfix)
%IF (CurDescription == null)
$(V_orcpcur)
%ENDIF
</B></FONT></TD>
</TR>
<TR><TD HEIGHT=5></TD></TR>
<TR><TD><BR></TD></TR>
<TR BGCOLOR="#E0E0E0">
```

```
<TD COLSPAN=5 ALIGN=center>
<FONT size=$(DONOTTRANSLATE_FORMAT_FONTSIZETEXT)>
<B>$(TXT_YOUCHARGED)</B>
%IF (ConvMultOrDiv == "")
<B>$(CurPrefix)$(FORMATTEDTOTPRICE)$(CurPostfix) $(V_orcpcur)</B>
%ELSE
<B>$(CurPrefix)$(FORMATTEDTOTPRICE)$(CurPostfix) $(CurDescription)</B>
 [$(ConvCurPrefix)$(CONVFORMATTEDPRICE)$(ConvCurPostfix)
 $(ConvCurDescription)]
%ENDIF
</FONT>
</TD>
</TR>
%}
 </TABLE>
 </CENTER>
 <BR>
%}
%MESSAGE{
100: {  No  Information Available. %} : continue
    default: { ERROR in DISPLAY_CHARGES_MerchantTax() %}:CONTINUE
%}
%}

%function(dtw_odbc) GET_CONF_MESSAGE() {
SELECT omornbr,ompaymthd,ompaydevc,pmentinst2,pmentinst4
FROM   ordpaymthd,merpayinfo
WHERE  paymerid=$(MerchantRefNum) and omornbr=$(order_rn)
%REPORT {
%ROW {
%if (V_ompaydevc == "OFF-LINE")
@DTW_assign(CONF_MSG, V_pmentinst2)
%else
@DTW_assign(CONF_MSG, V_pmentinst4)
%endif
%}
%}
%MESSAGE{
default: {<FONT COLOR="red">error occurred in
GET_CONF_MESSAGE()</font>%}:CONTINUE
%}
%}
```

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

%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT"></HEAD>
<TITLE>$(LongStoreName) [$(TXT_TITLE_ORDEROK)]</TITLE>
<BODY BACKGROUND="$(BodyImage)" BGCOLOR="$(BodyColor)"
TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
%IF (CurDecimalPlaces == "")
@DTW_assign(CurDecimalPlaces, "2")
%ENDIF
%INCLUDE "/CSPstoremodel/assign_priceseparators.inc"
<CENTER>
<TABLE width=530 CELLPADDING=4 CELLSPACING=0 BORDER=0 ALIGN="center">
<TR>
<TD ALIGN="center" VALIGN="center">
<FONT
COLOR=$(TextCol)><H$(DONOTTRANSLATE_FORMAT_FONTSIZETITLE)>$(TXT_TITLE_O
RDEROK)</H$(DONOTTRANSLATE_FORMAT_FONTSIZETITLE)></FONT>
</TD>
</TR>
</TABLE>
</CENTER>
@IS_SET()
%IF (($(PAYMENT_METHOD) == "SET") ||($(PAYMENT_METHOD) == "SETNV") )
@SET_TAXRULE_FLAG()
@SET_SHIPRULE_FLAG()
%ELSE
@GET_CONF_MESSAGE()
%ENDIF
@GET_SHOPPER_TYPE()
@GET_ORBILLTO()
@SHOPPER_INFO()
@SHOPPER_SHIPTO_INFO()
@DISPLAY_DETAILS_LIST()
@DISPLAY_CHARGES_MerchantTax()

%if (CreationFlag == "True")
@UPDATE_CREATION_ACCESS()
```

```
<center><a
href="/cgi-bin/ncommerce3/ExecMacro/ncadmin/sitemgr/creation_access.d2w
/report?merfnb=$env.merchant_id$">Click here to create your on-line
store!</a></center>
%endif

@DISPLAY_CUSTOM_NAVBAR()
</body>
</html>
```

Save this file and exit. This new macro will update the database with a flag granting store creation access if it has been purchased.

5. Stop and restart the Net.Commerce instance, administrator server and webserver as directed in *"Installing and Getting Started Guide"*, GC09-2808-01.

G.3 National language support

We have uncovered a problem with the use of non-US characters in the catalog editor, it will not accept any specific national characters. We have made a modification to the script that converts the catalog data to XML.

<inst. dir.>/NetCommerce3/Tools/public/javascript/convertToXML.js

```
function replaceSpecialChars(obj)
{
   var string = new String(obj);
   var result = "";
   var xtnd = 0;

   for (var i=0; i < string.length; i++ ) {
      if (string.charAt(i) == "<")       result += "&lt;";
      else if (string.charAt(i) == ">")  result += "&gt;";
      else if (string.charAt(i) == "&")  result += "&amp;";
      else if (string.charAt(i) == "’")  result += "&apos;";
      else if (string.charAt(i) == "\"") result += "&quot;";
/* This line converts all non-US ascii charaters to entities */
      else if (string.charCodeAt(i) > 127 )  result += "&#" +
string.charCodeAt(i).toString() + ";";
/**/
else result += string.charAt(i);
   }
   return result;
}
```

Appendix H. Special Notices

This publication is intended to help professionals who need to plan for and implement the IBM Net.Commerce Hosting Server on RS/6000. The information in this publication is not intended as the specification of any programming interfaces that are provided by Net.Commerce Hosting Server or Net.Commerce. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Net.Commerce Hosting Server Version 3.1 for more information about what publications are considered to be product documentation.

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 information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. 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

IBM ®	AIX	DB2
DB2 Universal Database	RS/6000	RISC System/6000
WebSphere	Net.Data	IBM Payment Server
SecureWay		

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  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 redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.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 Redbook 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.

Index

ITSO Redpaper Evaluation

Customizing Net.Commerce Hosting Server
REDP0022

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to `redbook@us.ibm.com`

Which of the following best describes you?
_ Customer _ Business Partner _ Solution Developer _ IBM employee
_ None of the above

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction __________

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

____

____

____

____

What other redbooks would you like to see published?

____

____

____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

____

____

____

____

Customizing Net.Commerce Hosting Server

REDP0022

REDP0022
Printed in the U.S.A.