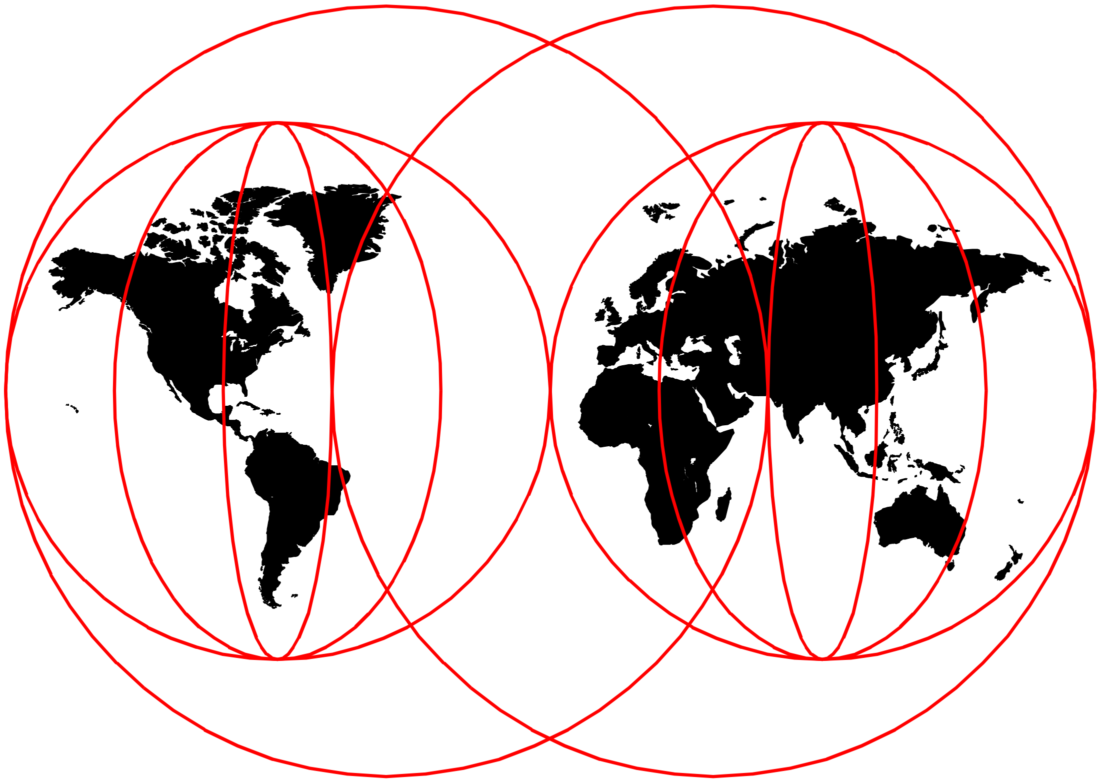
Net.Commerce V3.2 for AS/400: A Case Study for Doing Business in the New Millennium

Fant Steele, Ursula Althoff, Rui Fan, Lauren Hain,
Charles Haramoto, Shahar Mor, Lars-Olov Spångberg



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5198-00

International Technical Support Organization

SG24-5198-00

Net.Commerce V3.2 for AS/400: A Case Study for Doing Business in the New Millennium

July 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, "Special Notices" on page 533.

First Edition (July 1999)

This edition applies to V3.2 of IBM Net.Commerce for AS/400, Program Number 5798-NC3 and IBM Payment Server V1.2 for AS/400, Program Number 5733-PY1 for use with supported versions OS/400.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JLU Building 107-2
3605 Highway 52N
Rochester, Minnesota 55901-7829

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

Contents

Figures

Tables

Preface

Discover a practical end-to-end solution for doing business on the Web with Net.Commerce for AS/400. This redbook presents a case study that you can use as a guide for implementing a Net.Commerce site. The information in this redbook helps you plan, install, tailor, configure, and troubleshoot a Net.Commerce site by taking you through the implementation of a sample site. The intended audience for this redbook includes analysts or consultants that will sell, plan, or design Net.Commerce sites. It also targets the person that will build the Net.Commerce site.

The team that wrote this redbook followed the steps that a typical customer would consider when implementing Net.Commerce. The authors cover such topics as populating the Net.Commerce site with existing data, integrating with a backend system, and using payment collection. These areas are presented from the perspective of "What needs to be done to make this work". Please note that some knowledge of the AS/400 platform and Net.Commerce is assumed.

Note

This redbook is based on Net.Commerce for AS/400 V3.2 running on OS/400 V4R3. While most of the redbook still applies to Net.Commerce V3.2 running on later versions of OS/400, some specific instructions are only for V4R3. Chapter 9, "Setting Up SSL Using DCM" of this updated softcopy version has been modified to include DCM V4R4. If you are configuring or running Net.Commerce V3.2 on V4R4, please make sure you obtain Informational APAR II12011 and II12041. They contain important setup and WebSphere information. To access the APAR information, log on to:

`http://www.as400service.ibm.com/`

Click the **+** (plus sign) next to **Tech Info & Databases->Software Problems - APARS->All Info APARs by Release->Search**. Enter the APAR number, and click **Search**. The search should return the title of the APAR. Click the hyperlink by the title, and read the APAR information.

You should also check the readme file. To access the readme file, log on to:

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

Click **Support** in the left frame. In the right frame, support areas are listed. Click **IBM Net.Commerce** in the **Technical Library** area. Click **OS400** in the header bar. From here, you can view the readme file and the list of PTFs needed for different levels of the OS/400 operating system.

The Team That Wrote This Redbook

This redbook was produced by a team of AS/400 and Net.Commerce specialists from around the world working at the International Technical Support Organization Rochester Center.

Fant Steele is an Advisory ITSO Specialist for AS/400 in the International Technical Support Organization, Rochester Center. He writes extensively and teaches IBM classes worldwide on many areas of AS/400 communications technologies and e-business. He spent eight years as an instructor and developer for the AS/400 communications and programming curriculum of IBM Education and Training. Prior to joining IBM in 1989, he worked on S/36 to AS/400 code conversion, VM/MVS systems programming, and applications programming for the manufacturing industry.

Ursula Althoff is an AS/400 System Engineer working at IBM Midrange System Sales Technical Support in Germany. She has worked at IBM for 24 years. Her areas of experience include OS/400, application development, Internet Services on AS/400, and Net.Commerce for AS/400 Version 2.

Rui Fan is a Software Engineer from IBM AS/400 Partners In Development division in Rochester, MN. He has a year and an half of experience in the field of e-commerce on the AS/400 system. He has worked with IBM AS/400 Business Partners on designing and implementing AS/400 e-commerce solutions. His areas of expertise include Web development, Object-Oriented Programming, and e-commerce, specifically Net.Commerce.

Lauren Hain is an Application Integration Specialist in IBM Global Services Rochester Minnesota. He has 26 years of experience in hardware and software development. His areas of expertise include application development, Web development, and the customizing of e-commerce solutions using Net.Commerce.

Charles Haramoto is an IT Architect at IBM Global Services in Chile. He has been with IBM for nine years working in systems engineering, networking services, and consulting. Now he focuses on Net.Commerce and WebSphere technologies for implementing e-business solutions.

Shahar Mor is an AS/400 specialist working for an IBM Business Partner YUVAL in Israel. He has 12 years of experience in the AS/400 field. His areas of expertise include AS/400 TCP/IP and Internet connectivity, databases, PC connectivity to the AS/400, security, and client/server programming. Shahar teaches AS/400 classes for IBM Israel. He is also responsible for system services and technical support for YUVAL.

Lars-Olov Spångberg is an Advisory IT Specialist in IBM Global Services Sweden. He has worked at IBM for 20 years, which includes 19 years of experience in the S/38 and AS/400 field. His areas of expertise include IT security consulting, Internet services on AS/400, Firewall for AS/400, TCP/IP, and AS/400 performance.

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

Kris Peterson
Marv Kulus
Thomas Gray
Jenifer Servais
Marcela Adan
International Technical Support Organization, Rochester Center

Gaspare Latona
Janette Wong
The rest of the Net.Commerce for AS/400 Team
IBM Toronto

Daniel Luo
IBM Rochester

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 Redbook Evaluation” on page 549 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`

Part 1. Planning the Net.Commerce Site

The first part of this book introduces you to the concepts of e-business and Net.Commerce. Then, it explains the various aspects of planning your Net.Commerce site, including the infrastructure, site design and language considerations, integration with back-end systems, and payment collection. It also covers the tools and skills needed to complete the project.

Chapter 1. Introduction to e-business and Net.Commerce

Welcome to Net.Commerce for AS/400. This redbook describes an end-to-end solution showcasing the features of Net.Commerce for AS/400, as implemented with OS/400 Version 4, Release 3.

The Internet is becoming more common and, in many cases, essential to today's society. As a result, it is becoming important for businesses to be capable of exploiting the opportunities represented by the Internet. It is no longer sufficient to simply provide access to the Internet or perhaps even have a static presence. The real potential of the Internet lies beyond those early uses to actually conduct business transactions—conducting "commerce". The Internet-related technologies including TCP/IP, HTTP, browsers, and so on, are generally capable of three primary implementation scenarios. One is an *intranet*, which typically means the users that are connected together using the protocols are within an organization or business. A second use is connecting to the *Internet*, with typical functions such as internal users having the ability to access other Internet sites, as well as a Web presence. Finally, many businesses are establishing *extranets*, which use Internet technologies to connect to other businesses with whom they have a relationship.

Internet implementation is generally thought of as providing business-to-consumer value. Net.Commerce is IBM's premier offering that allows many businesses to quickly conduct consumer-to-business transactions through the Internet. It is a merchant solution that provides a framework to conduct business on the Internet in a secure and scalable manner. It supports both business-to-business as well as business-to-consumer environments. Net.Commerce works together with a relational database and Secure Web Server to give users and companies a simple and secure environment. Net.Commerce is designed to be scalable to meet the needs of the small to very large business. Merchants can take advantage of their existing operating environment and expand to larger systems as their electronic traffic grows. Figure 1 on page 4 gives a high level overview of the components that make up Net.Commerce.

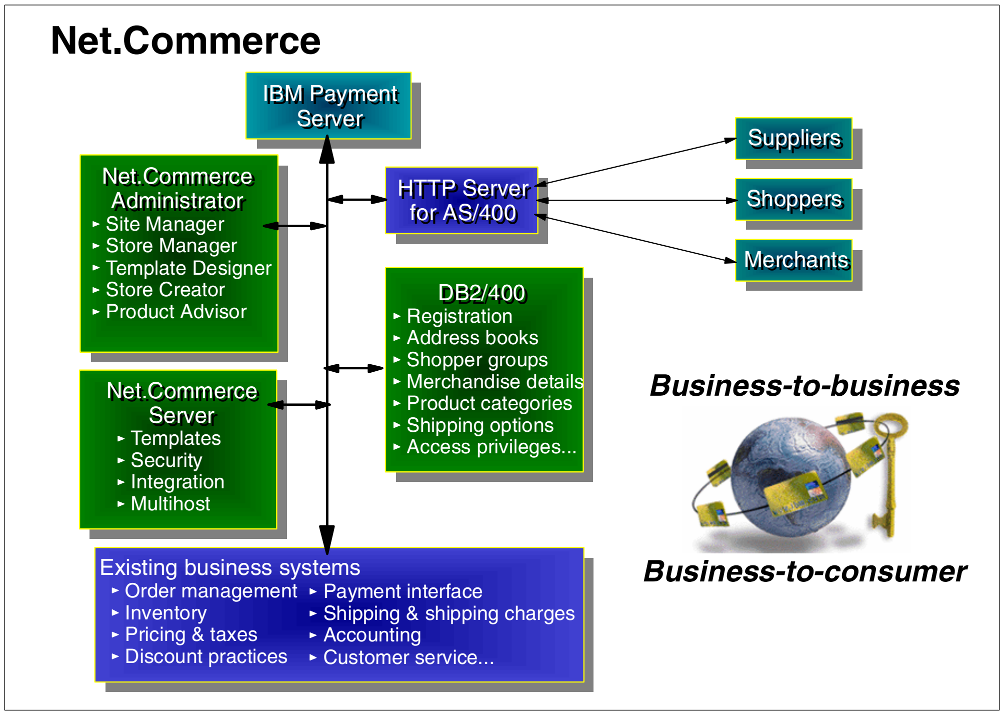


Figure 1. Net.Commerce Overview

The Net.Commerce system enables merchants to create electronic stores where they can sell their products and services globally over the Internet's World Wide Web (WWW). Using Net.Commerce, merchants can update their product information easily and tailor the way information is presented, and create "shopper groups" and offer their members special promotions or unique store views. They can also track demographic information that is provided by shoppers, and view purchasing statistics. Shoppers around the world can browse online catalogs of products and services, complete with descriptions and multimedia objects such as graphics, photos, and video and sound clips. They can place items in an electronic "shopping cart". Then, they can order them by using SET and an e-wallet or by providing credit card and shipping information. With Net.Commerce, shoppers can complete transactions from their own computers while avoiding crowds and checkout lines.

Net.Commerce is an application solution available from IBM to run on all IBM server platforms, including RS/6000 (AIX), S/390, PC servers (running Microsoft's NT), and the AS/400 system. Net.Commerce also runs on the Sun Solaris platform.

Before jumping right into Net.Commerce, a review of how the Internet-related technologies have achieved their current status as tools of choice for applications may be helpful. This provides a base on which Net.Commerce makes sense as the application of choice for business transactions. Two of the primary factors that have occurred are the evolution of client/server computing to Web technologies, and the maturation of technologies that enable business transactions to use the Internet as a business medium.

1.1 An Evolution to the Web

Over the past 10 years or so, an evolution in the computing paradigm has been occurring. The evolution has been going on even longer, but with regard to client/server computing, an evolution that began with the personal computer (PC) has come to dominate much of the data processing industry. When PCs were first introduced, they were largely a curiosity, in which only a select few people were interested. As their capabilities increased, so did the number of people using them, and eventually the mainstream IS professionals took notice. PCs quickly became an alternative for host dependent terminals, which allowed integration into the network, although the applications which they accessed remained largely unchanged.

Soon, however, applications that were the domain of the PCs and their users caught the attention of businesses. Many were "personal productivity" applications such as word processors and spreadsheets. This new breed of application was difficult, if not impossible, to replicate on host or centralized computers, and became increasingly important to the daily operations of businesses. The transition from terminal emulator to "workstation" was beginning to take hold. Some groups envisioned the transition would end with the complete elimination of the central or host computers, although we now know that vision was short-sighted.

Many true visionaries had a different view—one that allowed both the PC as a workstation and the central or host computer to maximize their respective values. This view held that applications should not be specific to a single system, but should work in concert on both types of systems—the PC workstation (client) and the central or host system (server). The tasks that required task-specific CPU-intensive functions such as graphical presentation would be done on the client. Meanwhile, the host would serve the data and (some) applications from a managed system. This client/server computing model was touted as the next great evolution of the data processing industry.

1.1.1 Client/Server Detour

Something happened on the way to client/server computing. As the early adopters began implementing applications in a true client/server model, they found that it was more difficult than envisioned. The difficulties included inconsistent splitting of the application between the client and server, lack of mature tools for application development, and less than expected, or more accurately, inconsistent performance between the various client and server systems.

Some of the difficulties were addressed by defining client/server models to help guide the split of the application. In the meantime, tools for development were improving more and more. Still, the actual process of successfully implementing and deploying a true client/server application remained difficult. Additional changes were required if this model was to become pervasive in the industry.

During this time of rapid change, as many "experts" tried to solve the client/server challenges, several attempts were made to simplify the client/server environment. One way to simplify a difficult task is to standardize the client/server platforms. Some progress was made, especially on the client side. Most businesses standardized their client hardware and software platform on "win-tel". This was the platform represented by Intel-based hardware personal computers (PCs) and Microsoft's Windows family of operating systems (Windows 3.1x and Windows 95).

Similar attempts at standardizing the server were never really universally accepted, though not for lack of effort. First came the demand for "open" systems, which varied so much in meaning that no one could really define "open". Some viewed an open server as one where the hardware vendor and software vendor were by definition two distinct entities (following the Intel-Microsoft model on the client). Others held that open was anything but IBM, which during this time was going through challenging business changes. After a while, the open label was put on systems running the UNIX operating system, giving hope to many that some standardization was finally occurring. What they did not realize was that there was so much variability among UNIX operating systems (there were and are dozens of variations). Even UNIX could not carry the open banner by itself. Smaller-scale efforts to define APIs or interfaces to systems made some progress as well. However, standardizing the server side of the client/server model remained elusive.

In the meantime, businesses kept running their operations. The specific technologies they used remained important, but were not the central issues that drove business decisions. Businesses were still hoping to accomplish client/server computing, but realized the process would be difficult. As such, they began taking small steps toward the goal of client/server computing, without a complete switch from their traditional processing models. These steps included getting their various systems to at least communicate with each other, sharing data, and in some cases, deploying basic client/server applications while steadily populating the desktop environment with PCs. These steps would become the foundation for client/server computing, but still more change would be required.

1.1.2 Parallel Web Development

During this time, a seemingly unrelated set of developments were occurring. A community of computer users, largely based in research and academia, were continuing to share data over a network. As the PC became popular in this community as well, the sharing of data began to evolve. The users found it more convenient to simply access remote data than to continually move it around their network. This was in part due to the limits of the day, which were primarily the speed of the network and the capacity of the individual systems. To more effectively work together in this arrangement, their network, or inter-connected network of systems, adopted a standard protocol.

This network, now evolved into the Internet and Transmission Control Protocol/Internet Protocol (TCP/IP), were the foundations on which one more piece was to be added. The missing piece was a way of dynamically sharing the information on various systems in the network. An application, including protocol, was developed to dynamically link information from these systems together. It was dubbed the Hyper-text Transfer Protocol Application (HTTP). The application was developed in a client/server model, with the client being responsible for end-user interaction and presentation management, and the server largely responsible for storing and transmitting data. The client was given a special name called a *browser*, while the server was simply referred to as an HTTP server. During this time, the network protocol (TCP/IP) emerged from the research and academic community into the commercial business world. In addition, the applications and protocols soon followed, including the newest one—HTTP.

The application/network combination was referred to as the *World Wide Web* (WWW or Web). In an almost fad-like fashion, people began to use it, play with it, experiment with its capabilities, and more. The client software—the browser—was of special interest. While the first browsers simply accessed and displayed information from a remote system, their limit was the style of presentation. They were text-based, much like host-dependent terminals. Since most client platforms of the day were fast becoming intelligent workstations such as PCs, the text-based limitation did not last. Seemingly overnight, an industry segment developed with a sole focus on the browser. The browser revolution was beginning.

Browser software quickly became a hot commodity, and the pace of development of browser-based tools, applications, and functions was unlike any environment up to this time. Even so, few would predict what was to happen next. Since the client/server development process was still bogged down, the opportunity presented by the browser was poised to revive it. Most people had given up on standardizing the server. Further standardization of the client to a specific application, the browser, changed the direction of client/server computing. Now application developers could use the browser as the generic or universal client, and would only have to focus on the server. Since the server application was already defined through HTTP, a quantum leap in simplicity in the client/server application development environment had seemingly occurred overnight.

The evolution is continuing, but clearly the Web and the browser to HTTP-server model of computing has surpassed the client/server application model to become a de facto standard in the data processing industry. Further evidence of this is the continuing development and maturing of application development tools for this specific environment.

1.2 Maturing Technologies

Many of the early applications developed for the Web environment were simple in nature, often providing access to publicly available information. Perhaps due to its roots in research and academia, these library-like uses were to be expected. However, with the whole realm of application development moving to this environment, the nature of the applications soon diversified. Businesses looked to this new environment to supplement and, in some cases, replace traditional applications. One example of this is the area of customer service. With the proliferation of PCs in the home, business found it viable to move some customer service functions to the Web environment, effectively providing for customer self-service.

A limiting factor to which applications were appropriate for this environment was security. Accessing publicly available information and perhaps some customer service applications were OK, but without security, specifically data privacy, other applications would not likely be appropriate for the Web. Prior to the Internet and Web phenomena, data privacy in the industry was a concern to only one company at a time as their applications and systems were deployed. With the Web being a public network, data privacy had now become a universal concern. Fortunately, since the browser had become the universal client, the solution to data privacy was quickly implemented. Existing data encryption technologies simply had to be incorporated into the HTTP browser to server protocol, and the application developers would individually not have to worry about data privacy. HTTPS (secure HTTP) is the protocol that was developed to provide data privacy over the Internet. One other area of maturing technology that continues to develop is the area of application development tools. The Web environment has given rise to a whole new development environment—JAVA. While the promise of the JAVA environment, where an application can be written once to run on a variety of hardware and software platforms, has long been desired, the Web environment has provided the momentum to make the promise a reality.

1.3 Ready for Net.Commerce

Finally, all the pieces were in place. The network, protocols, security, and development environment were mature enough to develop commercial applications. While many businesses develop their applications in-house, more and more businesses are purchasing applications. One such application, Net.Commerce, is specifically designed for the business-to-business and business-to-consumer retail environment through the Web. It requires the universal client, a browser, and provides all of the server-side application components. It can be described as a table-driven generic retail application, with the tables to be filled in by the businesses using it. It is highly customizable and scalable, and is lauded in the industry as perhaps the best retail application for the Web environment. The rest of this redbook describes the specific way in which this state-of-the-art application is implemented on the AS/400 system.

1.4 Additional Information

While it is important to know what is in this document, it may also be of interest to know what this document does not contain. Sufficient documentation exists for the base Net.Commerce product itself, and is not duplicated here. The Net.Commerce products are packaged with extensive online documentation, including both cross-platform and platform-specific reference manuals, and so on. Those sources should be used in conjunction with the material found in this publication.

1.5 What ShopITSO Is

To illustrate the entire process of creating an e-business Web site using Net.Commerce, we are going to use a fictitious store: the ShopITSO online store. In the following chapters, we describe how we designed and implemented the store.

Then, we explain how we connected it to a fictitious back-end system for real-time price consultation. We also describe the networking and payment considerations you need to take into account in order to build a secure e-business site and finally collect the payments from your customers.

Chapter 2. Planning: The Infrastructure

This chapter contains the information you must know to implement the infrastructure for your Net.Commerce site. Before you implement your Net.Commerce solution, you must carefully plan how you are going to connect to the Internet, protect your recourses, and connect your Net.Commerce server with your back-end system. You must also be sure that you have all of the necessary program products installed.

2.1 AS/400 Hardware Sizing

When deciding on the AS/400 model to be used, consider the complex nature of Net.Commerce command processing (CGI programs that may run, the database accessed, and so on). The system is likely to be doing much more than simple HTML page serving.

2.1.1 AS/400 Net.Commerce Hardware Requirements

At the time this redbook was written, the recommended minimum hardware requirements for running Net.Commerce alone on an AS/400 system is any RISC model of the AS/400 system capable of running OS/400 V4R3, with 350 MB of free disk space for program files and 96 MB of memory.

These are the requirements to install the Net.Commerce software and perform some limited shopping tasks. Testing in the laboratory has indicated that most customers need a minimum of 256 MB of memory running on an AS/400e server 170 with a CPW rating of 100 or more.

The minimum hardware requirements for operating a production site can only be determined through an assessment of your database volume and traffic requirements. You are advised to contact your IBM Sales or Service Representative for advice in this area.

A sizing tool is being developed to assist in hardware size determination and will be available from the Net.Commerce Web site in the AS/400 Download section for Net.Commerce V3.2. The Net.Commerce site is located at:
`http://www.software.ibm.com/commerce/net.commerce/`

Refer to the readme document at the Net.Commerce Web site in the AS/400 Download section for Net.Commerce V3.2 for the latest size recommendations.

2.1.2 Optional AS/400 Net.Commerce Hardware Requirements

If you plan to use the IBM Firewall for AS/400, you need an Integrated PC Server (IPCS) or Integrated Netfinity Server with two LAN adapters and 64 MB of memory.

2.2 AS/400 Net.Commerce Installation Requirements

Before you install AS/400 Net.Commerce on your AS/400 system, you must verify that both the AS/400 system and the administration PC meet the software requirements. This section takes you through the requirements.

2.2.1 AS/400 Net.Commerce Software Requirements

To run, configure, and administer the Net.Commerce server on the AS/400 system, you need two types of software:

- Licensed programs installed on your AS/400 system
- Software installed on your administrator PC

2.2.1.1 AS/400 Net.Commerce Software Requirements

The software requirements for AS/400 Net.Commerce V3.2 are:

- OS/400 Version 4 Release 3 (5769-SS1)
- Digital Certification Manager (5769-SS1 Option 34)
- TCP/IP Connectivity Utilities for AS/400, V4R3 (5769-TC1)
- Licensed Program Product (LPP) 5769-DG1, V4R3, which includes:
  - IBM HTTP Server for AS/400
  - IBM Web Sphere Application Server 1.1
  - IBM Net.Data for AS/400
- One of the following IBM Cryptographic Access Provider products to use Secure Sockets Layer (SSL)
  - 5769-AC1 - 40-bit
  - 5769-AC2 - 56-bit
  - 5769-AC3 - 128-bit
- AS/400 Net.Commerce (5798-NC3)
- For Java Servlet Support, IBM AS/400 Developer Kit for Java (5769-JV1)
- Qshell Interpreter (5769-SS1 Option 30)

Note: You *need to have* all of the latest PTFs for the above listed products applied on your system. You can obtain the latest PTFs either by applying the latest cumulative package, fix pack, group PTFs, and individual PTFs, or by ordering the PTFs directly from your AS/400 service representative. For more information, consult these sites on the Web:

- For links to various AS/400 products and their PTFs, go to: `http://www.as400.ibm.com/misc/map.htm`
- For a link to the list of PTFs available for the IBM HTTP Server for AS/400 and IBM Web Sphere Application Server 1.1, go to: `http://www.as400.ibm.com/http`

2.2.1.2 AS/400 Net.Commerce Administrator PC Software Requirements

You configure and administer the AS/400 Net.Commerce site through a Web browser on a PC. The administration PC requires the following software:

- Configured and operational TCP/IP support
- A Web browser that support HTML frames, Java Script and Java 1.1.4, for example Netscape Communicator 4.06 or higher, 32-bit version

2.2.1.3 Browser Requirements for Shoppers

To shop in a store that is created with Net.Commerce, shoppers can use any browser that supports the following features:

- SSL
- Java and Java Script

- Tables and frames
- Cookies

2.2.2 Optional AS/400 Net.Commerce Software Requirements

Depending on what you intend to do, you must install one or more of the following software products either on the AS/400 or on your workstation. There may be other software products than those listed here that are useful for your Net.Commerce implementation. Here are a list of some software products that can help you with your Net.Commerce implementation:

- IBM AS/400 Client access (5769-XW1 and 5769-XD1)
- AS/400 Operations Navigator, which is a part of AS/400 Client Access
- Firewall for AS/400 (5769-FW1)

  If you install the Firewall for AS/400, you also need Integration Services for IPCS (5769-SA2).
- IBM Payment Server 1.2 for AS/400 (5733-PY1)
- SSL Payment program, for example I/NET Merchant 400
- DB2 Query Manager and SQL Development Kit for AS/400 (5769-ST1)
- ILE RPG for AS/400 (5769-RG1)
- VisualAge RPG for Windows (5763-CL2)
- VisualAge C++ for AS/400 Windows95/NT (5716-CX5)
- AS/400 Native C++ compiler PRPQ (5799-GDW)
- AS/400 Toolbox for Java (5763-JC1)
- Domino for AS/400, V4.6.2 or higher, if you want to use Domino e-mail and the discussion database from Net.Commerce

Note: We recommend that you have the latest PTFs for the above listed products applied on your system. You can obtain the latest PTFs either by applying the latest cumulative package, fix pack, group PTF, or by ordering the PTFs directly from your AS/400 service representative. For access to links to various AS/400 products and their PTFs, go to the following site on the Web:
`http://www.as400.ibm.com/misc/map.htm`

Tools that can be useful when you build the Net.Commerce site include:

- Net Object Fusion
- Claris
- FrontPage
- The Net.Commerce Template Designer
- Net.Data design tool (for Windows NT)
- XML
- The Net.Commerce Mass Import Utility
- The Net.Commerce Database Cleanup Utility
- And many other software products

2.3 Network Planning

It is important that an infrastructure is in place before any implementation of the AS/400 Net.Commerce server take place.

2.3.1 Network Security

When connecting to an untrusted network, you must ensure that your security policy provides you with the best protection possible. A firewall certainly represents a large portion of your total security solution. However, because a firewall is only the first line of defense for your network, you must ensure that your security policy provides additional coverage.

2.3.2 Security Policy

A security policy is a written document that defines the security controls that you institute for your computer systems and the risks that these controls are intended to minimize. A security policy also defines what actions should be taken if your security controls are breached.

The *most important rule* that your security policy should express is: *Anything that is not explicitly permitted, should, by default, be denied.* In other words, automatically disallow any actions that you do not specifically allow. This ensures that new types of attacks are not likely to get past your defenses. However, you may have no knowledge of them and have nothing in your security controls to defend specifically against them.

A security policy contains rules, such as who can access certain services or which services can be run from a given computer. The policy also contains information about what processes and controls are instituted to enforce these rules. If you are connecting to the Internet, your security policy should stipulate that you install and use a firewall to control access to and from the Internet. Figure 2 illustrates the major components of an Internet security policy.

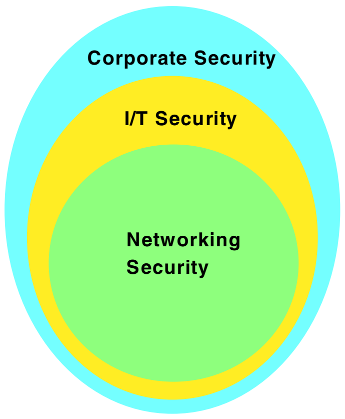


Figure 2. Components of an Internet Security Policy

Once you create a policy, you must ensure that it is put into effect. This may involve establishing more restrictive password rules, installing and running virus protection software, holding classes to educate users on security rules, and so on.

2.3.2.1 Policy

It is important that your implementation follow your company security policy. This section provides some examples from different parts in an I/T Security Policy, which is a part of the Corporate Security Policy. Normally, an I/T Security Policy has several pages. What we provide here is only a small extract from an example of an I/T security policy. In the following example, we call the company: Mycompany.

General I/T Security Policy Statement

- A Mycompany Information System (IS) is any information or telecommunications system owned, leased, or operated by Mycompany.
- Mycompany will implement at least the minimum security requirements as identified in this policy, to protect IS resources and information (non-sensitive and sensitive data) processed, stored, or transmitted by Mycompany ISs. Based on risk management, they may apply additional safeguards to provide the most restrictive set of controls (privileges) that permit the performance of authorized tasks (principle of least-privilege).
- Sensitive information in Mycompany ISs must be safeguarded against unauthorized disclosure, modification, access, use, destruction, or delay in service.
- All ISs that process, store, or transmit sensitive information must be accredited.
- Connectivity is prohibited between Mycompany ISs that handle sensitive data and any other systems or networks not under Mycompany authority, unless formally approved by an appropriate Mycompany Accrediting Authority.
- All Mycompany ISs are for Mycompany business only and users have no expectation of privacy while using these resources.
- All persons must comply with these policies who use, manage, operate, maintain, or develop Mycompany ISs, applications, or data.

Internet Services Policy

Mycompany owned or controlled ISs may only access the Internet through Mycompany approved gateways. This limitation means that Mycompany owned, controlled, or authorized computer equipment, regardless of its location or means of connection to any network or system, may not be used to access the Internet, directly or indirectly unless such connection is through a Mycompany approved Internet gateway (firewall). While the configuration of some networks make it technically possible to access the Internet without going through an approved gateway, such access is not authorized.

Exceptions to this policy must be approved in writing by the Director of the Telecommunications Department.

Table 1. Security Policy Planning Worksheet

Prerequisite Checklist (All answers should be Yes before you proceed with the Installation)	**Answers**
Do you have an I/T Security Policy, and a Network Security Policy in place?	

2.3.2.2 Security Service

The National Institute for Standards and Technology (NIST) defines five major security services. To completely protect your network, your security policy should address each of these areas as well:

Authentication	Assurance that the resource at the other end of the session is really what it claims to be.
Access Control	Assurance that the resource requesting access to data or a service is authorized to have access to the data or service.
Integrity	Assurance that the information that arrives is the same as the information that was sent.
Confidentiality	Assurance that sensitive information is not visible to an eavesdropper. Encryption is the best way to ensure confidentiality.
Non repudiation	Assurance that a transaction can be proven to have taken place — also called *accountability*.

2.3.3 Network Security Objectives

Although the network security objectives that you develop depend on your particular situation, there are some general objectives to consider:

- Protect your resources, including:
  - Your Internet servers
  - Your internal network, workstations, and systems
  - Your data
  - Your company's image
- Provide your customers with safe Internet transactions. Ensure that the following conditions are in place:
  - Communicating parties can identify each other (authentication).
  - Unintended parties cannot read information exchanged between parties (confidentiality).
  - Unauthorized parties cannot alter data (integrity).
  - Participating parties cannot repudiate transactions (accountability).

For more information regarding Network Security read *AS/400 Internet Security: IBM Firewall for AS/400*, SG24-2162, and *IBM Firewall for AS/400 V4R3: VPN and NAT Support,* SG24-5376.

2.3.4 Operating System

Before any installation, you have to verify that you have at least OS/400 Version 4 Release 3 (5769-SS1) or later installed. We recommend that you have the latest PTFs for the operating system applied on your system. You can obtain the latest PTFs either by applying the latest cumulative package, fix pack, group PTF, or by ordering the PTFs directly from your AS/400 service representative.

2.3.5 TCP/IP Configuration

You must have a configured and operational TCP/IP environment on your AS/400 system. For more information about configuring TCP/IP on your AS/400, please read *TCP/IP Configuration and Reference*, SC41-5420.

2.3.6 Server Placement

Placing the Net.Commerce server behind the firewall provides both a high level of security for the private-secure network and more protection for the Net.Commerce server. The firewall blocks all access to the internal network from the Internet.

2.3.6.1 Placing the Net.Commerce Server behind the Firewall

When you place your Net.Commerce server behind the firewall, you gain the following advantages:

- The firewall protects the Net.Commerce server. You do not depend on the ISP router for protection of the Net.Commerce server.
- You can use the firewall logging function to detect and recover from attacks on the Net.Commerce server.
- The Net.Commerce server and production data are on the same side of the firewall, which may make it easier for you to update the Net.Commerce server with production data.
- You can use the same AS/400 system to run the firewall Integrated PC Server or Integrated Netfinity Server and run the Net.Commerce server.

By placing the Net.Commerce server behind the firewall, more protection is provided for the server. Filter rules are added to allow only certain types of traffic to be passed to the Net.Commerce server. Any other packets are discarded by the firewall. Even if the ISP does not filter packets, your firewall protects the Net.Commerce server.

The firewall can also provide logging of packets. If you choose to use this feature, you receive a log that contains information about packets that are accepted and forwarded, and packets that are discarded. These logs can be used to determine if someone has been attacking your network. The logging features must be set up before they can be used.

By having the Net.Commerce server and the production systems protected by the firewall, you can easily use built-in tools, such as Distributed Relational Database Architecture (DRDA) or File Transfer Protocol (FTP), to move data between systems without having to modify the firewall. This allows access to existing data and systems when implementing Internet-based applications.

One system running OS/400 at V4R3 or later is needed to support Network Address Translation (NAT) on the IBM Firewall for AS/400. This same system can be used as the Net.Commerce server because the firewall protects the secure interface from attack.

Table 2. Prerequisite Planning Worksheet

Prerequisite Checklist (All answers should be Yes before you proceed with the Installation)	**Answers**
Is your OS/400 V4R3 or later?	
Is TCP/IP Connectivity Utilities for AS/400 (5769-TC1) installed?	
Is Digital Certification Manager (5769-SS1 Opt. 34) installed?	
Is IBM HTTP Server for AS/400 (5769-DG1) installed?	

Prerequisite Checklist (All answers should be Yes before you proceed with the Installation)	Answers
Is IBM Cryptographic Access Provider (5769-AC1, AC2, or AC3) installed?	
Is AS/400 Net.Commerce V3.2 (5798-NC3) installed?	
Did you verify that the most current PTFs available are installed?	
Is TCP/IP configured in your AS/400 system (including IP interfaces, routes, local host name and local domain name?	
Does your Net.Commerce administrator workstation have a Web browser that support HTML frames, Java Script and Java 1.1, for example Netscape 4.04 with Java plug-in or Netscape 4.5?	

2.3.7 Firewall Planning

You should consider NAT in your planning process. For general planning considerations regarding IBM Firewall for AS/400, refer to *Getting Started with IBM Firewall for AS/400,* SC41-5424, and the redbooks *AS/400 Internet Security: IBM Firewall for AS/400,* SG24-2162, and *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376. IBM makes frequent updates to the AS/400 Firewall home page. Check the latest tips and updates at:

`http://www.as400.ibm.com/firewall`

Consider the following points when planning to implement a firewall using the NAT function:

- Determine the servers and ports to which access is allowed. Notice that you can use the same public address (for example, the non-secure port of the firewall) in multiple MAP settings, if you map to different ports. IBM Firewall for AS/400 basic configuration automatically creates filter rules and MAP settings if you specify that you have public HTTP and HTTPS servers behind the firewall. You need to configure filter rules and MAP settings to enable other public servers behind the firewall.
- The firewall non-secure port IP address and the public IP addresses assigned to servers behind the firewall must be on different subnets. This does not apply to the special case where the IP address assigned to the public servers is the same as the non-secure port of the firewall.
- Determine the ISP router configuration. Plan to configure the ISP router correctly.

  If the To_addr is the same as the firewall's non-secure IP address, then no routes are required. If the To_addr is some other address, then the router must be configured so that it routes traffic for the To_addr using the firewall's non-secure IP address.
- You must install the DB2 for AS/400 Query Manager and SQL Development Kit (5769-ST1) licensed program product if you want to convert firewall logs to DB2/400 tables and use interactive SQL to build views of your log data.

Table 3. Firewall Planning Worksheet

Questions about Your Firewall	Answers
Is your Firewall installed and configured?	

2.3.8 Connection Planning

There are several point that you must consider when you are planning to connect to the Internet:

- Type of connection
- Line speed
- Domain name
- Number of IP addresses

Some of the parameters depend on the amount of workload and net traffic you plan to have. If you do not have an Internet connection, you have to contact an Internet Service Provider (ISP). Normally, the ISP can help you to register a domain name and give you IP-addresses.

Table 4. ISP Planning Worksheet

Questions about Your Internet Service Provider (ISP)	**Answers**
Have you already selected your Internet Service Provider (ISP)?	
Is your connection to the ISP installed and verified?	
Is your ISP responsible for configuring the router that connects your perimeter network to the ISP?	
Has your public domain name (*mycompany.com*) been registered with the InterNIC?	

2.3.8.1 Planning for SSL

To use Secure Sockets Layer protocol (SSL), you need a digital certificate assigned by a Certificate Signer, for example, IBM World Registry or VeriSign. To be your own certificate authority, you need to have Digital Certificate Manager (DCM) installed. For more information about the SSL, read *TCP/IP Tutorial Technical Overview,* GG24-3376.

Digital Certificate

A digital certificate identifies a user or a system and is required before SSL can be used. Once a server has a digital certificate, SSL-enabled browsers, such as the Netscape Navigator, can communicate securely with the server using SSL.

A digital certificate is issued by a certificate authority (CA). CAs are entities that are trusted to properly issue certificates and have controls in place to prevent fraudulent use. The certificate authority charges a fee for issuing a certificate.

Some examples of universally recognized Internet certificate authorities (CA) include:

- Thawte
- VeriSign
- US Postal Service
- AT&T
- MCI

For testing purposes or for applications that will be used exclusively in an intranet environment, you may issue digital certificates using an intranet certificate authority. The AS/400 system with *Digital Certificate Manager* (DCM) can act as an intranet certificate authority.

You can configure your AS/400 system as an intranet certificate authority. Digital Certificate Manager (DCM) is a Web-browser based administration facility that allows you to create, manage, and use certificates within an enterprise and with partners of an enterprise. You can use DCM to request digital certificates from Internet Certificate Authorities such as VeriSign and Thawte.

DCM allows you to create your own intranet certificate authority (CA). You can then use the CA to dynamically issue digital certificates to servers and users (client certificates) on your intranet. When you create a server certificate, DCM automatically generates the private key and public key for the certificate. You can also use DCM to register and use digital certificates from VeriSign or other commercial organizations on your intranet or the Internet.

Digital Certificate Manager is option 34 of OS/400 (5769-SS1, option 34). You must install this option to use DCM. DCM is a link in the AS/400 Tasks page, which runs in the *ADMIN HTTP server instance. Therefore, you must have installed IBM HTTP Server for AS/400 (5769-DG1) and use it to access DCM. In addition, you must install IBM Cryptographic Access Provider licensed program product (5769-AC1, or AC2, or AC3) to create certificate keys. These cryptographic products determine the maximum key length permitted for cryptographic algorithms on your AS/400 system. Government export and import regulations determine which version is available in your country. To use all the options available in DCM, you must have *SECOFR and *SECADM authority.

Table 5. SSL Planning Worksheet

Questions about Your SSL Implementation	Answers
Have you already selected your Certification Authority?	
Have you received your digital certificate?	

2.3.9 Planning for SET

If you plan to use Secure Electronic Transactions (SET) on your Net.Commerce server, there are some points that you must consider. For planning information regarding how to setup SET, see Chapter 6, “Planning: Payment Collection” on page 91.

2.4 Server Integration

You can have more than one Web server instance running on your AS/400 system. All HTTP servers are originally set up to use the well-known port 80. What you have to think about is that Net.Commerce must use port 80.

To avoid a conflict on this port, only let the HTTP server instance for Net.Commerce use port 80. Then, configure the other instances to use a different port, or use another IP address for the other Web servers instances. Either inform your users about the different ports or IP addresses that you are using, or make links from the server listening on port 80 to the other Web servers. A good idea may also be to use virtual hosts from the server listening on port 80 to redirect to the other Web servers instances listening on other ports.

If you use the same port but different IP-addresses, use the HTTP directive *BindSpecific On*, in *all* your HTTP configuration members, to bind to just one IP address.

For more information about configuring the HTTP configuration member on your AS/400 system, please read *IBM HTTP Server for AS/400 Webmaster's Guide*, GC41-5434.

2.4.1 HTTP Server

When you configure your Net.Commerce server, it creates an HTTP instance for you. You can have more than one HTTP server instance running on your AS/400 system. Avoid port conflicts by using different ports or different IP addresses for the HTTP servers.

2.4.2 Domino Server

The Domino Web Server is part of Domino for AS/400. The Domino Web server can serve Domino databases and also HTML files. If you are already familiar with Domino on other platforms, there is no difference between the AS/400 system and others.

For information about setting up Domino on the AS/400 system, please read *Lotus Domino for AS/400 - Installation, Customization, Administration*, SG24-5181. Or, consult the Lotus books *Installing and Managing Domino for AS/400*, Part No. 12999, and *Extending the Domino System*, Part No. 12953.

Chapter 3. Planning: Site Design Considerations

The design phase of any successful implementation of Net.Commerce is key. As with all types of projects, the design phase is usually the most critical phase after the requirements have been gathered. It is often stated that for every hour spent in design, you save several hours during the implementation phase. As a result, the importance of the design phase cannot be understated.

The main inputs to the design phase are the functional requirements produced during the requirements gathering phase. After all, it is the foundation upon which all of the other phases are built and developed. In this phase, you should cover all the aspects that are needed to implement an e-business application that map your business process and rules.

Please note that because every project is different, this discussion does not have the capacity to offer low-level methodologies on how to resolve each of these issues. Its purpose is only to save some time and effort by presenting issues that you may have otherwise overlooked.

The following sections present a number of general considerations to think about when migrating to an e-business. For each of these points, you have to decide how you want to do your e-business. On the basis of these decisions, you can proceed with the design.

As a result from the requirements and design phase, your design documents should be sufficiently specific to allow the implementation team to take the design documentation and begin the implementation phase. The important parts of these documents are the navigation flow for the whole e-business application and a detailed description of the functionality of each screen in the e-commerce site.

Important

With the Net.Commerce product, you receive a large range of sample shops with different business behavior. When one of these samples meets your requirements, you can use this as a basis for your shop. Perhaps you can use this shop unchanged.

3.1 General Considerations

This section presents important considerations to keep in mind before and during migration from an existing business to an e-business. The intent is not to cover every possible business process that exists. We assume that you understand the fundamentals inherent in running a typical business. Instead, the primary focus is on specific e-business issues that may not be apparent at first glance. Think of this part as a checklist of points that you need to consider and address.

3.1.1 Audience and Scope

The first question that should be answered in the initial planning stages is: *With whom do I do business?* As a direct result of this question, you must also ask: *To what extent do I do business?* Both questions need to be answered as soon as possible to ensure the resulting e-business is feasible.

The following sections highlight the differences in these processes and examine other details that could affect e-business planning.

3.1.1.1 Business-to-Business Customers

In the case of business-to-business customers, you sell products to other resellers. Your customers are running their own businesses, so efficiency is a key factor to consider. Most likely, they know exactly what products they want, so they want to place their orders as quickly as possible and return to their own work.

They should be provided with a product line and online catalog specific to their needs and a user interface that is direct and easy to use. In addition, each business has its own processing systems (payment, order, shipping, and so on). It is your responsibility to ensure that you can correctly interface with them. The key is speed and efficiency.

3.1.1.2 Business-to-Consumer Customers

In the case of business-to-consumer customers, you sell products to customers who will personally use them. Therefore, your customers are not as pressed for time. In fact, they may not know exactly what they want when they first come to browse your virtual store. Here, the key is a good shopping experience.

The most obvious first step is to have an attractive user interface. However, be sure to balance aesthetic appeal with functionality (no one likes to sit waiting for a page to load all of its images). In fact, most customers find that functionality is what truly makes for a good shopping experience. Similar to business-to-business customers, business-to-consumer customers should be given personalizations based on factors such as customer profiles, personal preferences, and buying patterns. Together with other components, such as easy site navigation and straightforward payment processing, this ensures that customers will return to your site for future purchases.

3.1.1.3 Geography

Consider where your business is currently located. If it is a small- or medium-sized business, it may be in a single town or in a few states. If it is large, it may span an entire country or in some cases the whole world.

Now, consider where your e-business will be located. Most likely, your site will be accessible to anyone in the world who has an Internet connection. The next question is whether you should conduct business as you normally would or take into consideration this vast new group of potential customers. If you opt for the latter, then you should consider these issues:

- **Language**

  How many languages will your site support? Will multiple languages affect any of your business processes (such as customer service)?

- **Currency**

  How will you handle currency conversions? Particularly, will you have support for the euro?

  How you can handle dual currency for the euro support? See the implementation of the Net.Commerce sample Euromall. You install this mall during the configuration phase of the Net.Commerce instance. See Figure 124 on page 178 in 12.1, “Creating New Net.Commerce Instances” on page 173.

- **Taxation**

  How will you handle sales tax, local tax, national tax, excise tax, and duties? Will there be circumstances that result in tax exemptions?

- **Shipping**

  How do you handle shipping charges to various countries? Will you limit your shipping range, or will you be able to ship anywhere in the world?

- **Products**

  Will your product line differ by geography? Will customers even have the option to choose by geography? Will product prices be calculated and displayed based on geography?

- **Advertising**

  Who is your target audience (local or global)? Will advertising be perceived differently by different cultures?

Again, please realize that this is by no means a complete list of every geographical consideration that exists. However, it should be enough to start you moving in the right direction. You can read more about this in Chapter 4, “Planning: Language Considerations” on page 85.

3.1.1.4 Store or Mall

Consider your existing product line. Do all of your products belong to the same category, or do they differ widely? Perhaps because of physical or financial constraints, your store seems more like multiple stores, only under the same roof. Although with an e-business, these constraints are much less stringent. Therefore, it is possible to set up an online mall consisting of multiple stores that would seem more logical to the shopper.

However, this introduces some issues that should be noted. For example, each store should have its own look and feel to give the customer a sense of really being in a mall of individual stores. Also, each store will attract different shopper groups that could possibly affect the way sales, discounts, and other related entities are carried out. As always, there are numerous factors to consider, and the final decision rests on careful planning.

3.1.1.5 Competitors

In your existing business, perhaps your main competitor is the local convenience store across the street. Then again, maybe it is a Fortune 500 company across the ocean. When you move to an e-business, it is any person or organization that happens to be selling the same items that you are. Aside from industry giants that are immediately recognizable, how will customers be able to differentiate a well-established, quality company from a struggling startup? Most likely they will not unless there are other means to differentiate them. Specifically, factors such as service (both online and offline) and product quality will help to sway the customer’s decision. In the end, a customer would definitely be more willing to pay a little more for these amenities.

Another issue to consider is what happens when you have both online and offline competitors. Consider the following scenario:

> A company that sells music CDs has a physical store in a very expensive neighborhood. By virtue of its location, the store must have prices that closely match those of its local competitors. However, this company also has an online store that sells the exact same CDs. Here, the prices must be lowered because they seem much too high compared with competitor prices on the Web. Can the company justify selling the same CDs for different prices?

This example illustrates just one of the many complexities that must be addressed when migrating to an e-business.

3.1.2 Shoppers

From a shopper's viewpoint, the basic store procedures are very similar for both a physical store and a virtual store. Therefore, in both cases, shoppers will typically enter the store, browse through the inventory, select a product, engage in a checkout process, and leave the store. However, there are some specialized shopping issues that require specific considerations.

3.1.2.1 Preferred Shopper Groups

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

In your e-business, there are many parallels to the preferred shopper group schema. Instead of regular and one-time-only shoppers, an e-business normally has registered and guest shoppers. Similar to the application for a “club card,” registered shoppers must supply information such as name, billing address, shipping address, and phone number. In return, they are given (or personally create) a unique login name and password to your site.

This functions as their “club card” and places them in your virtual store's preferred group. By logging in each time they come to your store, they will have access to specials and discounts that guest shoppers will not be able to access.

3.1.2.2 Personalization

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

In your e-business, personalization can be accomplished in a much more efficient manner. Because your system is constantly monitoring the products customers place in their shopping carts, it can instantaneously offer related products or accessories for existing products. Personalization can also go beyond just the sale of products. Registered shoppers could also possibly change the look and feel of the virtual store to suit their preferences. However, it is important that the personalization does not become too intrusive. For example, shoppers can become very annoyed if a related product is pushed onto the screen every time they access their shopping cart. An alternative may be to offer one or two products during the checkout process. As always, it is important to ensure that the customer has a good shopping experience.

The Net.Commerce e-business application provides you ways to handle both types of shoppers. The guest shopper (a one-time-only shopper) and the registered shopper, where the personal information of the customer is registered in the database. For registered shoppers, you can create shopper groups. For shopper groups, you can assign special discounts, and special category and product templates to personalize the shopper experience and to define promotional offers.

3.1.3 Products

For the most part, the products you sell in your existing business will not change when you migrate to an e-business. The only aspect that changes is how the products are sold. However, there are a few issues that you should at least briefly consider before you put your products for sale online.

Your product range should be subdivided into categories and subcategories. This provides your online shoppers a general idea of your supply. With this hierarchy, you can guide the shoppers through your catalog. See 3.2, “Planning the Product Catalog” on page 31, for more information about how you can structure the product catalog. Another approach may be to split up your products and categories in different stores in a mall (see 3.1.1.4, “Store or Mall” on page 23).

3.1.3.1 Pricing

A product can have multiple prices associated with it, depending on factors such as sales, discounts, and taxes. You have to decide the best way to handle these prices.

For sales, it must be determined whether to store two different prices in the database for each sale product (for example, regular price: $10, sale price: $8) or to have a set deduction amount (for example, $2 deduction on sale items). There must also be some mechanism to cause sales to expire at some future time.

For discounts, it must be determined whether to have a discount by quantity (for example, $5 off the total if you buy 10 widgets), a discount by customer (for example, registered customers get a 5% discount on widgets), a discount by category (for example, 10% off all widgets in this category), or a combination of discounts.

For taxes, it must be determined whether to include the tax in the product price or to have it added separately. Also, there is the question of whether certain tax exemptions will be based on the type of product or geographical considerations.

Net.Commerce provides you with a flexible way to offer special discounted prices. The discounting functions provide five different implementations for customizing discounts. You can also specify whether discounting will be based on quantity, cost, or weight, and whether it will be calculated based on total item orders (cumulative) or on a series of ranges. By default, the discounting implementations are set to the store scope, meaning that each individual merchant can choose an implementation. All the information you need to use the following discount possibilities is stored in several tables of the Net.Commerce database. To define a discount, you need to consider the following factors:

- What will be discounted? Discounts can be applied to complete orders or to products matching a discount code. For product discounts, the calculation can be performed against all products matching the code or separately for each individual product.
- To whom will it be offered? Discounts can be offered to all shoppers or specific shopper groups.
- When is it valid? Discounts can have a start and end date and time. For example, you may wish to have a one-day-only sale.
- What is the scale determined by? Each discount has a scale (a number of ranges which have a discount rate). The scale ranges can be based on quantity, weight, or currency amount.
- How is the discount value applied? Each rate can be applied as either a total value, per unit of the scale or as a percentage. Multiple rates can be added cumulatively.
- Should the discount be calculated on a suborder before tax and shipping charges are applied or on the complete order after tax and shipping charges. Local laws may determine this decision.

To learn how you can implement and work with the Net.Commerce discount possibilities, see Chapter 6, "Database Implementation of Discounts," in the *Net.Commerce Utilities* handbook. This guide is included with the Net.Commerce product. You can find this book in the directory: `/QIBM/ProdData/NetCommerce/html/MRIxxxx/ncbooks`, where xxxx is your language code, for example, 2924 for the English language. You can also find a short overview in 3.4, “Working with Net.Commerce Discounts” on page 38.

3.1.3.2 Subsets

The next question is whether to put up for sale online your entire existing product line or only a subset of it. This is an especially important consideration if you decide to keep your existing business operating in addition to your e-business. Perhaps certain products in your inventory appeal to a global audience, where others only appeal to a local audience.

Perhaps there are too many online competitors for a certain product, so it would only make sense to sell that product in your physical store. Or maybe it would not make sense to sell certain products online. Again, these are just a few suggestions to start you thinking on the right track.

3.1.4 Payment Processing

In your existing business, payment processing most likely is straight forward. Customers simply pay for products at the cashier, using various forms of payment. When payment is approved, the purchase is completed. In an e-business, payment processing is a bit more complicated.

You can use the Secure Electronic Transaction (SET) protocol, which provides shoppers with a high degree of security for payment card transactions over open electronic networks. SET support in Net.Commerce is integrated transparently with the rest of the Net.Commerce software and shares a common database. Net.Commerce can manage both SET orders from customers with a SET wallet such as IBM Consumer Wallet for Windows 95 and non-SET secure server transactions from customers who cannot or do not wish to use SET. This is called the *Merchant Originated Payment.* See 6.2.1, “Merchant Originated Payment” on page 96, for more information about this method.

Merchant Originated Payment allows the merchant to receive credit card information through any mechanism, for example, through the store’s online order forms. When the form is submitted, the credit card information is encrypted using SSL. It is then passed to the acquirer using regular SET messages, through the IBM Payment Gateway for further processing like request capture. For more information about payment processing with SET, see Chapter 6, “Planning: Payment Collection” on page 91.

3.1.5 Order Processing

In an e-business, order processing is more like a mail-order company. Inventory must be checked, shipping providers must be contacted, customers must be notified, and many related events must occur. Thus, the process required to fulfill a customer order has many considerations.

3.1.5.1 Inventory

When an order is placed, the customer expects the order to arrive in a reasonable amount of time. But what if the customer orders a product that is out of stock? This is an inventory handling issue that needs to be addressed.

One possibility is to have a mechanism to send low-inventory warnings. For example, if the inventory of widgets dwindles to five units, an alert should be sent that requests more to be manufactured or acquired.

An alternative may be to allow back orders. If an order is made for a product that has zero units in stock, the system should recognize this and warn the shopper. The shopper should be given the option of back ordering the item (realizing delivery will be delayed), selecting another comparable product, or canceling the order.

3.1.6 Shipping

Once an order is placed, payment has been authorized, and inventory has been checked, it is time to ship the product to the customer. But how will the shipping process occur? There are some important considerations to examine in the area of order shipments.

First, who will ship the orders? This will certainly depend on to whom you plan to ship. If you have a global audience, your shipping provider should be able to reach every corner of the world. Another significant issue is interfacing with your provider. Due to the nature of e-business, most processes are designed to be automated. Therefore, it is your responsibility to make sure that your systems can correctly interface with the shipping provider’s system.

Second, where will the orders be shipped? At first glance, the answer seems obvious—to the customer address. However, the customer’s address may actually consist of a billing address and a shipping address. One of the primary reasons for this feature is to allow a customer to send a gift. For example, Mr. A will have the bill sent to his address, but Mrs. B will have the dozen roses sent to her address.

In addition, there should be some means of validating addresses to make sure they actually exist before products and bills are shipped to them.

3.1.7 Notification

When should customers be notified that their order is on the way?

Again, this seems like an easy question to answer, but it should be approached with caution. Some may think that as soon as the shopper clicks the Submit button on the payment screen, a message saying “Your order is on its way” should be displayed. More accurately, the message should say, “Your order has been received and is being processed...you will be notified when it is confirmed."

After the order is checked against existing inventory and the shipping provider has confirmed shipping of the item, the customer should be notified about the status of the order. They should be notified preferably by e-mail or phone. You should be sure to include the date when the shipment will be done. This process should be so automated that it can take place in a matter of minutes (or even seconds) and the customer feels that the store interactions are efficient.

3.1.8 Order Status

The customer has just been notified that his or her order has been confirmed and it should arrive in two weeks. Invariably, in a week and a half, the customer wants to know the status of the order. In an e-business, customers should be able to check their order’s status 24 hours-a-day, 7 days-a-week at their own convenience and as many times as they want. This is another step in ensuring a good shopping experience for the customer, even after the initial sale has occurred. Perhaps your back-end system does the order fulfilment, so there should be information from your back-end system to your e-business system about the status of the order. This information can be used in the e-business system to change the order status.

3.1.9 Security

In an e-business, global accessibility is the norm. Therefore, security is of extreme importance to ensure that both the customer and the business are not affected adversely.

3.1.9.1 Customer Security

In a typical online shopping experience, customers often release sensitive information such as their home address, home phone number, and credit card number. By doing this, they are non-verbally stating that they trust you to handle this information with utmost confidentiality. Without a reasonable sense of security, it would be foolish for anyone to make transactions online.

The most common way to create a secure environment is by using the SSL (Secure Sockets Layer) protocol to *encrypt sensitive data* that is being transmitted from the shopper. This method of security has become so prevalent that every major browser now supports SSL sessions. For more information about planning for the SSL protocol, see 2.3.8.1, “Planning for SSL” on page 17. For information about how you can assign the use of the SSL protocol to a Net.Commerce command, see 13.17, “Assigning SSL Protocol to Net.Commerce Commands” on page 264.

SET is another security protocol that is used for secure payment processing. The SSL protocol is a prerequisite for using SET. For more information about this requirement, see Chapter 6, “Planning: Payment Collection” on page 91.

3.1.9.2 e-business Security

Your e-business security is just as significant as customer security is. The first security issue that most people think about is hackers. What is the best way to keep sensitive data away from people who are out to damage it?

One way is to set up a firewall between the world and your most sensitive data—namely the contents of your database. It is also imperative to limit accessibility to your intranet or servers from outside locations. For more information, see 2.3.1, “Network Security” on page 12.

3.1.10 Disclaimers and Store Policies

In your existing business, you have rules and regulations to govern what actions should be taken in specific circumstances, especially irregular ones. Because every business is different, these store policies vary from one store to the next. In an e-business, the general policies that apply to your existing store should also apply in this environment. It is important that a list of these policies is always available for the customer to read, and it should be highly visible on the site. This will ensure that your customers always know how your site conducts business and any legal information that may affect them. Here are some policies that should be common to every e-business:

- **Late deliveries**

  When should a customer be notified if a delivery will be late? Should they be compensated in these cases?

- **Incorrect deliveries**

  Should the customer receive a refund or store credit? Should the correct delivery be express mailed to the customer at your expense?

- **Out-of-stock products**

  Should the customer be able to back order? Should the customer be given a slight discount when the product is in stock again?

- **Returned products**

  Was the product unsatisfactory, or did the customer just change their mind? Does the customer want a refund or exchange?

- **Canceled orders**

  Should there be a time limit between when the order was placed and when the customer requested cancellation? Should the customer be required to pay all shipping costs?

3.1.11 Customer Service

Depending on how large your current business is, you may have an entire customer service department answering phone calls at a help desk or directly assisting customers in your store. In an e-business, customer service differs quite a bit with both improvements and drawbacks.

3.1.12 Existing Methodologies

Perhaps your business is already using computers for a great deal of business processing and data exchange. For example, you may be using some form of electronic data interchange (EDI) to send unfulfilled orders to your supplier.

The question is what should be done with these existing business processes when you migrate to an e-business. Read Chapter 5, "Planning: Integration with the Back-End Systems" on page 87, for more information.

3.1.13 Data Transfer

Perhaps your current business tracks existing product information in a database. One very significant step is to plan how to transfer that information into the databases for your e-business. There are many methods for accomplishing this. The appropriate method strongly depends on your existing database configuration.

A second factor is the correctness of the data. In any mass transfer of data, there is the likelihood for some sort of error. Again, because this data is crucial to the success of your e-business, it must be meticulously inspected for any faults or flaws. See Chapter 15, "Importing Business Data into Net.Commerce" on page 319, for techniques that can be used to import your existing data to the Net.Commerce database.

You should also plan for how changes in your existing database (your back-end system) are transferred to the e-business database. For example, when you add a new product in the back-end system, this information should also be populated into the tables of your Net.Commerce system. Our solution for doing this is described in 17.3.1, "Product Information Synchronization" on page 393.

Also changes (new orders, for example) from the e-business system must be transferred to your back-end system. For an example of how to do this, read 17.4, "Requesting Capture upon Order Fulfillment" on page 398.

3.1.14 Performance

Just like customers of your existing business, your e-business customers expect prompt, efficient service. To ensure that this criterion is met, consider the issue of performance.

The greatest factor affecting your site's performance is your intended audience. *Will your site be visited by just a few visitors a day or a few hundred?* Once this is determined, the appropriate hardware and software can be selected.

As a general rule, it is a good idea to overestimate the number of hits that you expect your site to receive so that any unexpected spikes in traffic can be managed. Overall traffic will also be affected by how much bandwidth you dedicate to your site.

No matter how sophisticated your hardware and software may be, sometimes issues inherent to the application's design are the sources of performance problems. In most cases, optimizations can help to alleviate many of these difficulties. For example, graphics and image optimizations can greatly speed up site navigation and loading. Use the caching functionality for graphics and images.

Finally, code optimizations can also significantly enhance performance and processing speed.

During the planning phase, it may also be advantageous to develop a set of performance benchmarks that you can measure your application against later in the project’s life cycle. This may not seem important in the planning phase, but it is always a good idea to document the standards you expect your application to meet before you actually start building it. These benchmarks may change during the course of the project, but it is helpful to compare them against your first impressions because they are often times the most accurate.

3.1.15 Tools

Another technical consideration is to have a set of online tools to help you perform various tasks. For example, it is very common to have reporting tools that calculate and display useful statistics about your site. These statistics may include details about the number of hits, the number of visitors, or even the number of registrations that your site received during a particular period of time.

Tools that can help you to develop the e-business site can be found in Chapter 7, “Planning: Tools to Build the Site” on page 103.

3.2 Planning the Product Catalog

Organizing products into categories allows shoppers to easily find what they want out of thousands of items for sale. Product categories provide an effective structure for your product line. They also lay out pathways for shoppers to navigate through the online store, starting at the home page, and ending at the product page.

This section shows you a sample of how a product catalog in Net.Commerce is planned. This should give you an idea of how to define the structure and rules that you will use to import your data into Net.Commerce from your production data.

Important

Define your rules about how you will structure your products into the Net.Commerce schema. You should do this before importing your data from a production system to the Net.Commerce databases.

For information about how you can import your data into this structure, see 15.2.2, “Mass Import” on page 321, and 13.7, “Creating the Product Catalog” on page 203.

Categorization is the process by which we define categories that group products with other similar products and create a hierarchical structure of categories so that all products appear below a single root.

To create product categories, first arrange the products in a hierarchy, or inverted tree. The tree begins at a general category (called the *root*), and branches out into increasingly specific sub-categories until it cannot be further divided. Each lowest level category, which contains only products, is a *leaf*. Within the structure, a category is the parent to the categories immediately below it, and a child of the ones above. This structure is stored in the database. The Web pages that display the catalog to the shopper can be built automatically using this data.

Figure 3 shows you an overview of the product catalog structure. It uses the following elements to build a structure:

Top or Root Category	The parent of all categories and products in the store.
Category	Group of subcategories. Usually products are not assigned at this level.
Subcategory	Group of products. Can have multiple parents.
Products	An instance of a commodity or service offered by a store. Products can belong to multiple parent categories. Many products consist of multiple items which are variations of the base product.
Item	A variation of a product, uniquely identified by one or more attributes or features such as size or color.

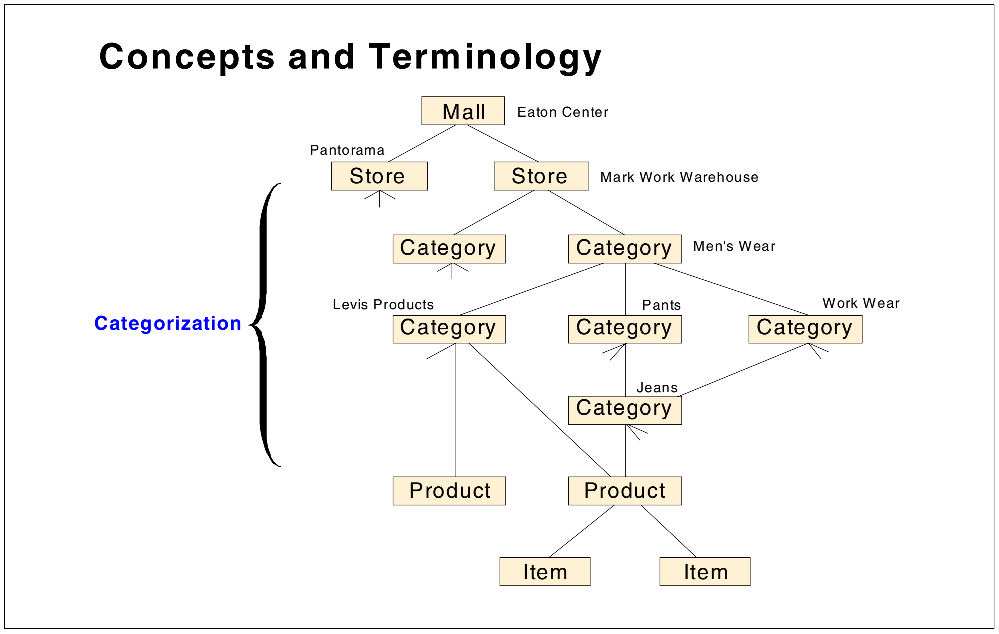


Figure 3. Structure of the Product Catalog

Do not make the tree structure too deep. The deeper the structure is, the more levels the shoppers have to traverse to find the product for which they are looking. This can lengthen the overall shopping process and may result in a shopper abandoning the search and leaving the site. As a general rule, you should avoid a depth greater than three categories under a store.

3.2.1 Category Structure

Now let us look at how the categories are implemented in Net.Commerce. Figure 4 shows how categories can be structured:

- **Root Category (1)**

  The top level of the store. All of the other product categories fall beneath it. It is the parent of Category 2 and 3.

- **Category (2)**

  Represents the product lines divided into main categories similar to the departments in a store. It is a child of Category 1 and a parents of Sub-Categories 4.

- **Category (3)**

  Represent the product lines divided into main categories similar to the departments in a store. It is a child of Category 1 and a parent of Sub-Categories 4 and 5.

- **Sub-Categories (4) (5)**

  The departments divided into types of products. Both are children of Category 3. Sub-Category 4 is also a child of Category 2.

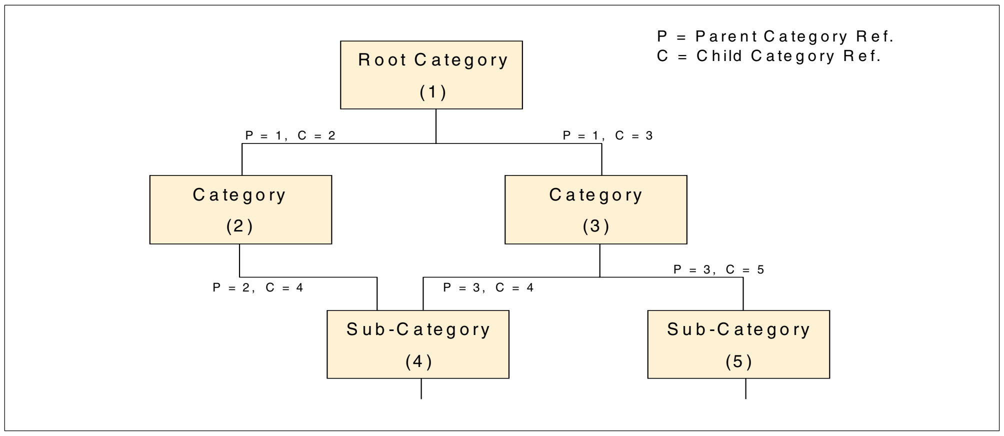


Figure 4. Category Relationships

The Product Categories form, accessed from the Product Categories icon in Store Manager, allows the creation of the category tree structure for the product line to customize navigation within the store.

Category properties such as names, descriptions, or images that can be shown in the online catalog can be modified. Categories can be moved or copied to effectively manage the category tree. When the product line changes, it may be necessary to move a category to another parent in the tree. It may be necessary to allow shoppers to access a category from more than one parent, so it appears in different routes. See 13.7, “Creating the Product Catalog” on page 203, for an example of how we built our product catalog. This is represented in the Net.Commerce database as described here:

Category database table (CATEGORY)
Contains information that describes the product categories and subcategories for each store. Each row describes one category.

Category Relationships database table (CGRYREL)

Defines the parent/child relationships between the categories and subcategories for each store. This information is used to structure the product categories that are presented to shoppers. Each row describes one relationship.

3.2.2 Product Structure

The product structure consists of these elements:

Product An instance of a commodity or service offered by a store. Some products are standalone and have no variations but others consist of multiple similar items with some different features.

SKU (Stock Keeping Unit) Number
An identifier for each item sold by a merchant. Often has a particular structure to relay information about the product without physically looking at the product.

Item A variation of a product. The different variations are distinguished by one or more features or attributes.

Attribute Represents one feature or property of a product or item, for example, color or size. Attributes have an associated value, for example color=blue or size=15m.

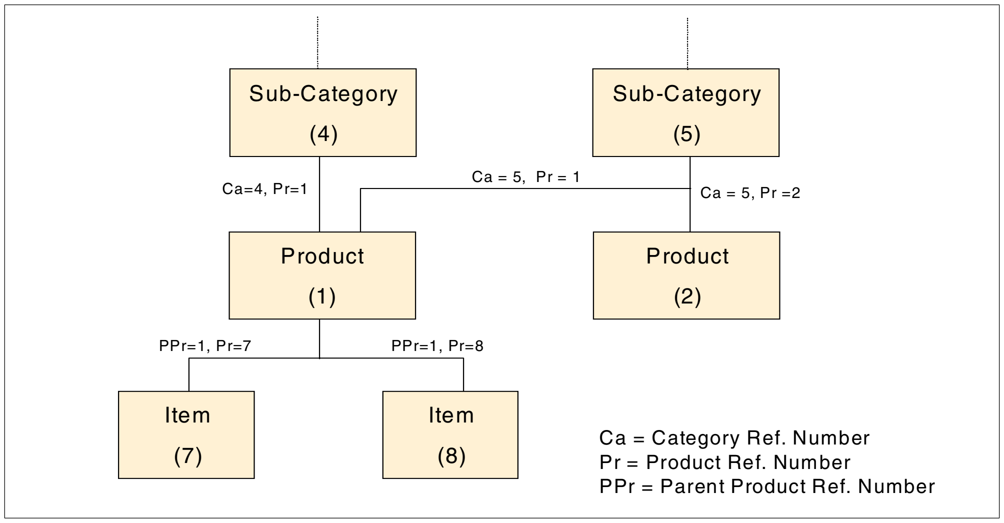


Figure 5. Product and Item Relationships

Note the following points:

- **Item (7) (8)**

  Instances of Product 1 with specific attributes. Each item has a SKU number. Children *only* of Product 1.

- **Product (1)**

  Describes information common to all of its children. *Product (1) cannot be purchased*. Child of Categories 4 and 5.

- **Product (2)**

  A product with no items. Has a SKU number. Child of Category 5.

This is represented in the Net.Commerce database as described here:

Category/Product Relationships table (CGPRREL)
: Defines the relationship between the products and their parent categories. Each row describes one relationship.

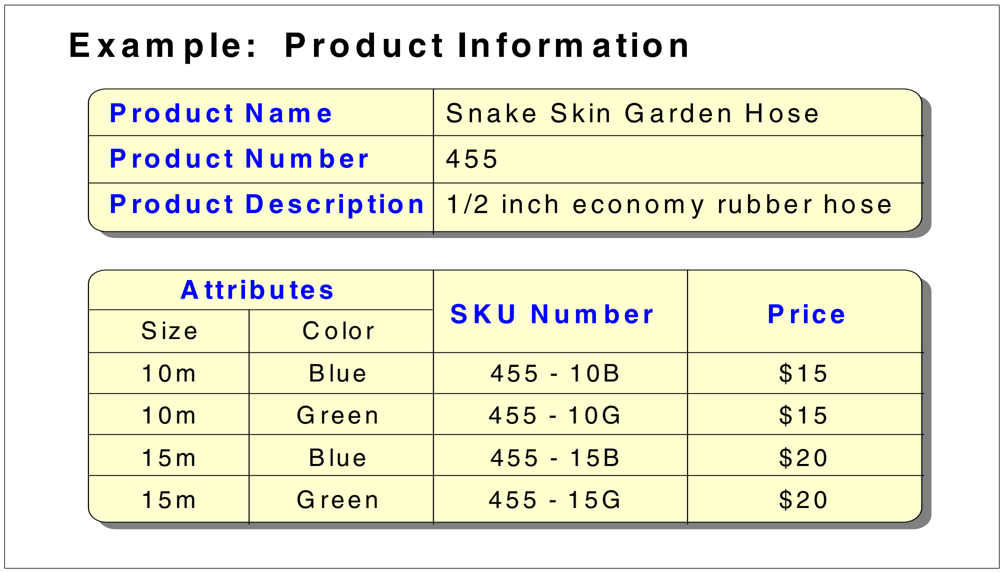
Product table (PRODUCT)
: All products and items available at a store. Items are rows in this table that point to their parent product in this table. The table contains the descriptive information for products.

Product Prices per Shopper Group (PRODPRCS)
: Each row in this table describes a product price for a shopper group. It is possible to have several prices for one product for several shopper groups. If the shopper group field is blank, this is the assigned price for normal customers.

Product Distinct Attribute (PRODDSTATR)
: Contains the names of all the distinct attributes for products.

Product Attribute (PRODATR)
: Associates attribute names with attribute values.

Figure 6 shows you a hierarchy of a sample product, which has four items with attributes.



Example: Product Information

Product Name	Snake Skin Garden Hose
Product Number	455
Product Description	1/2 inch economy rubber hose

Attributes		SKU Number	Price
Size	Color		
10m	Blue	455 - 10B	$15
10m	Green	455 - 10G	$15
15m	Blue	455 - 15B	$20
15m	Green	455 - 15G	$20

Figure 6. Sample Product with Items that Have Attributes

3.2.3 Planning Product Descriptions

There is also one other important point to think about with products. In our sample Shop ITSO store, we have only small text for the two long description fields in the product table. We imported this text from our back-end system table BELDSC (see Table 12 on page 205).

In your store, you may want to have detailed descriptions. Because the content of these fields is used to generate information that is displayed using a browser, you can import HTML formatted text into these two long description fields. To control the format for this description, you can use the HTML tags for such elements as colors, line brakes, tags for building lists or tables, and so on. Plain text can also be placed in the fields, and it will display as simple text.

To import this HTML formatted text in the product long description fields, you can use the Mass Import function when you first populate the Net.Commerce database. You may also want to put in simple descriptions to start and then change the description fields later as a second step.

For information about how you can change this text for a single product, refer to 13.10, “Using Product Long Description Fields” on page 214.

Another approach is to use the PRURL field in the PRODUCT table. You can use the content of this field as a URL link to another document (page). To learn how you can use this field to show additional description information about the product, see 13.11, “Using the Product PRURL Field” on page 218. When you use this URL, plan to create the HTML files to which this URL points and where it is stored to define the right URL value that is then stored in the PRURL field. The URL value should be imported to the PRODUCT table also through the Mass Import function. This means you have to define logic about how to build the Mass Import file to import the right URL value for the appropriate product.

Important

In the planning phase, you should consider how to build the descriptions for your products. Use a tool to create the HTML source files for these descriptions. Also, consider in which file these descriptions are stored. Build logic about how to create the Mass Import file to import the descriptions from these files to the product table description fields. Plan the use of the PRURL field in the same way. Use the Mass Import function.

3.2.4 Planning Category and Product Templates

Category and product templates are *Net.Data macros*. These macros are used as templates to create the dynamic Web pages. In these macros, you use SQL queries to get the required data from the database, for example the product description, the name of the image for this product, and so on. The resulting data from the SQL query is converted into the HTML format (a Net.Data function). This can then be shown in the browser. Also, all static information for these templates is defined in these Net.Data macros. You can assign different macros with different content to products or categories.

You can use Net.Data templates for special shopper groups. This gives you the ability to present special shoppers another style of product and category pages with content that is different from what a regular shopper sees.

In the Shopper Group Category Template table (CATESGP), you assign the name for the corresponding category Net.Data macro (template). In the Shopper Group Product Template table (PRODSGP), you assign the name for the corresponding category Net.Data macro (template).

The Net.Commerce commands `CategoryDisplay` and `ProductDisplay` use the assigned catalog or product Net.Data macro template to generate and display the content of the pages.

To learn how you can assign a Net.Data macro template to a category or product without the Mass Import function, see 13.8, “Assigning Templates” on page 206.

Note

If you plan to have more than one template for products or categories, you should define the logic to place the correct values in the template path and name fields in the PRODUCTS and CATEGORIES tables during the design process. Build the Mass Import file using this logic and use the Mass Import function to update the tables. For more information, see 15.2.2, “Mass Import” on page 321.

3.3 Images and Multimedia Files

Certainly you want to show your products, a logo, or other pictures in your Web pages. Perhaps you want to integrate such multimedia files as sound, moving pictures, and so on. You have to decide which software you will use to create these files. The media files are in-line, which means that they appear in the page that the shoppers see. Sound, animation, and video files tend to be large, so use them with discretion.

Note: Not all browsers support these in-line media files. Consequently, only shoppers who use a browser that can play media files will be able to see them.

Images and other media files are stored on the AS/400 server system in the IFS (integrated file system). The path and file name of the image for a product is assigned in the Net.Commerce database PRODUCT table in the PRFULL and PRTHMB fields. The path and file name of the image for categories is assigned in the Net.Commerce CATEGORY database table in the CGFULL and CGTHMB fields.

When you create your images for the products and for the categories, think about a rule for how you will name these images. This will be very helpful later when you import the data from your production system to the Net.Commerce database. You can then easily define a rule, for example, for how the Mass Import file (see 15.2.2, “Mass Import” on page 321) should be created to fill the path and name for the image in the above mentioned Net.Commerce database tables. Also, if you populate the NetCommerce database using your own program (if this is your decision), a rule for naming the images can be very helpful.

Note

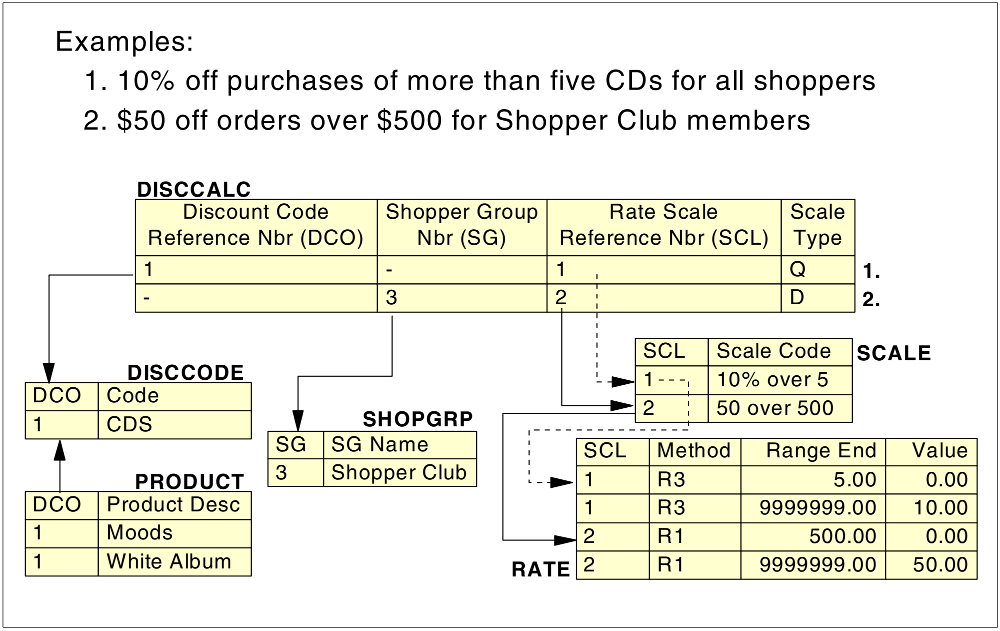
Name the image based on a field with a value that is unique for each product in your production system. For example, if the product number for the product is 4712123, name the image L4712133.gif for the large image and T4712133.gif for the thumbnail image.

3.4 Working with Net.Commerce Discounts

Net.Commerce provides you with a flexible way to offer special discounted prices. The database tables that define discounts are:

- **DISCCODE** — This table lists the discount codes that are assigned to the products. Each product can have a single discount assigned. Use this table to define a discount code as being used with products rather than orders. To discount at the product level, this discount code is placed in the records of the products it applies to in the PRDCONBR field of the PRODUCT table.
- **DISCCALC** — This table lists the discount calculation scale and type for each discount code. Each discount code can have one calculation per shopper group. A discount calculation with a discount code NULL identifies an order discount rather than a product discount. The scale type determines whether the ranges in the scale are based on Quantity (Q), Weight (W), or Dollar amount (D).
- **SCALE** — This table lists the rate scales that can be applied to a discount calculation. A scale consists of a unique reference number (a code up to 15 chars) and a description.
- **RATE** — This table lists the individual rates for a scale. Each scale can have multiple rates. A rate consists of a rate method, a range end, and a rate value. There are six computation methods, which are explained in Figure 8 on page 39. R1 uses the value as the discount amount, R2 uses the value as a unit discount amount, and R3 uses the value as a percentage of the total amount. The first range starts at 0 and *includes* the range end. The second range starts at the first range end + 0.0001.

Figure 7 shows you an example of how to define the tables for two different discount calculations.


Examples:

1. 10% off purchases of more than five CDs for all shoppers
2. $50 off orders over $500 for Shopper Club members

DISCCALC

Discount Code Reference Nbr (DCO)	Shopper Group Nbr (SG)	Rate Scale Reference Nbr (SCL)	Scale Type	
1	-	1	Q	1.
-	3	2	D	2.

SCALE

SCL	Scale Code
1	10% over 5
2	50 over 500

DISCCODE

DCO	Code
1	CDS

SHOPGRP

SG	SG Name
3	Shopper Club

PRODUCT

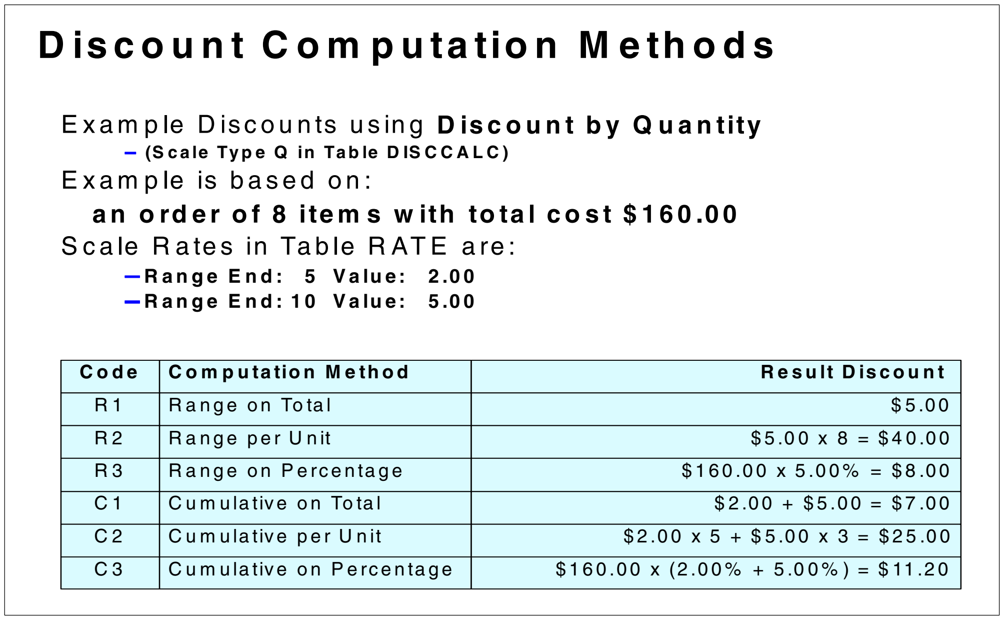
DCO	Product Desc
1	Moods
1	White Album

RATE

SCL	Method	Range End	Value
1	R3	5.00	0.00
1	R3	9999999.00	10.00
2	R1	500.00	0.00
2	R1	9999999.00	50.00

Figure 7. Discount Definition in Net.Commerce Tables

Figure 8 shows the different resulting discounts that are calculated when you use the Discount by Quantity (Q in table DISCCALC) scale type and six different computation methods. Notice that the same *range end* and *values* compute different results depending upon which method is applied. In some cases, the value refers to a monetary value, while in other cases (R3 and C3), the value refers to a percent value.



Discount Computation Methods

Example Discounts using **Discount by Quantity**

- **(Scale Type Q in Table DISCCALC)**

Example is based on:

an order of 8 items with total cost $160.00

Scale Rates in Table RATE are:

- **Range End: 5 Value: 2.00**
- **Range End: 10 Value: 5.00**

Code	Computation Method	Result Discount
R1	Range on Total	$5.00
R2	Range per Unit	$5.00 x 8 = $40.00
R3	Range on Percentage	$160.00 x 5.00% = $8.00
C1	Cumulative on Total	$2.00 + $5.00 = $7.00
C2	Cumulative per Unit	$2.00 x 5 + $5.00 x 3 = $25.00
C3	Cumulative on Percentage	$160.00 x (2.00% + 5.00%) = $11.20

Figure 8. Example Discount Calculation

Order cost calculation is performed by the Net.Commerce OrderDisplay command using a number of PROCESS tasks in the following order:

- **GET_SUB_ORD_PROD_TOT** — Calculates the cost per sub-order (shipping address)
- **GET_SUB_ORD_PROD_TAX_TOT** — Calculates the tax per sub-order
- **GET_SUB_ORD_PROD_SH_TOT** — Calculates the shipping per sub-order
- **GET_ORD_PROD_TOT** — Calculates the total cost of all sub-orders
- **GET_ORD_PROD_TAX_TOT** — Calculates the total tax of all sub-orders
- **GET_ORD_PROD_SH_TOT** — Calculates the total shipping of all sub-orders

To implement a discount function, override the default PROCESS task GET_SUB_ORD_PROD_TOT, or GET_ORD_PROD_TOT, or both with one of the five supplied overridable functions (discount functions) per PROCESS task. The built-in functions for these tasks *do not* support discounts.

To enable discounts by sub-order and have the tax and shipping charges calculated based on the *discounted total*, assign a discount function (OF) to the GET_SUB_ORD_PROD_TOT task, such as:

- `GetSubOrderProductTotalWithDiscountPerDiscCode` — Calculates a discount for each discount code in a sub-order. Does not support order discounts.
- `GetSubOrderProductTotalWithDiscountPerDiscCodePerProduct` — Calculates a discount for each product in a sub-order that has a discount code. This can

produce a different result than "PerDiscCode". Does not support order discounts.

- `GetSubOrderProductTotalWithDiscountPerOrder` — Calculates the discount based on the total sub-order cost.
- `GetSubOrderProductTotalWithDiscountPerDiscCodePlusPerOrder` — Calculates a discount for each discount code in a sub-order. A further discount can be applied based on the total sub-order cost.
- `GetSubOrderProductTotalWithDiscountPerDiscCodePerProductPlusPerOrder` — Calculates a discount for each product in a sub-order that has a discount code. A further discount can be applied based on the total sub-order cost.

If you want discounts calculated per order and have the tax and shipping charges calculated based on the *undiscounted total*, then assign a discount function (OF) to the GET_ORD_PROD_TOT task, such as:

- `GetOrderProductTotalWithDiscountPerDiscCode` — Calculates a discount for each discount code in an order. Does not support order discounts.
- `GetOrderProductTotalWithDiscountPerDiscCodePerProduct` — Calculates a discount for each product in an order that has a discount code. This can produce a different result than "PerDiscCode". Does not support order discounts.
- `GetOrderProductTotalWithDiscountPerOrder` — Calculates the discount based on the total order cost.
- `GetOrderProductTotalWithDiscountPerDiscCodePlusPerOrder` — Calculates a discount for each discount code in an order. A further discount can be applied based on the total order cost.
- `GetOrderProductTotalWithDiscountPerDiscCodePerProductPlusPerOrder` — Calculates a discount for each product in an order that has a discount code. A further discount can be applied based on the total order cost.

Each merchant can use one (or none) of the first five functions, and one (or none) of the second five functions to enable discounting.

Use Site Manager Task Management to override the built-in function with one of the previously mentioned discount functions. For information about how you can do this, see 19.4.7, “Assigning the Overridable Function” on page 434.

3.5 Planning Caching Facilities

Plan to implement the cache functions with Net.Commerce to avoid unnecessary workload for your Web server. There are two ways that you can use cache functions in a Net.Commerce e-business application:

- Use Net.Commerce caching functions

  With these functions, product and category pages, which are dynamically created by the `ProductDisplay` and `CategoryDisplay` commands, are stored in the nc_cache directory. This directory exists for each instance. The path is: /QIBM/UserData/NetCommerce/instance/<*instance_name*>/nc_cache.

  These commands retrieve information from your database, and display the information as an HTML page that has been generated from a Net.Data macro. If your product and category information has not changed since a page was last viewed, then it should not be necessary for the page to be dynamically re-created the next time a shopper requests it.

If you change information in the database (such as a product price), you want to make sure that the page in the cache will be deleted. You also want to be sure that the page will be dynamically regenerated the next time the product is viewed by a shopper and pick up the new information. Deleting the "stale" pages is done for you by the *synchronization daemon*.

Keep in mind that when you integrate a back-end system, as we did in our ShopITSO example to get the price information, this refresh mechanism should also be integrated. Refer to 17.3.2, “Integration with Net.Commerce Cache Mechanism” on page 393, to see how we did this and to obtain more information about the Net.Commerce caching functions.

- Store your static HTML files, images (GIF files), and all other static files in the AS/400 HTTP Web server local cache

  By keeping your most frequently served files loaded in the server's memory, you can improve your server's response time for those files. For example, let us say that you load your server's home page into memory at startup by adding it to the cache list. In this case, the server can handle requests for the page much more quickly than if the server had to read the file from a disk.

  To learn more about using the AS/400 HTTP Web server local cache, see 13.14, “Using the HTTP Web Server Cache for Static Pages” on page 234.

3.6 Summary Checklist — Side Design Considerations

This section presents a checklist of the major topics discussed in 3.1, “General Considerations” on page 21. The checklist also includes some additional topics that are specific to an e-business implementation. Use the checklist either as a quick review of everything discussed or as a reference of considerations to use during requirements gathering.

I. Audience and Scope

A. *Business Types*

1. Business-to-Business
   - Efficiency most important
   - Online catalogs specific to each business
   - Fast, easy-to-use navigation
   - Interface with other businesses existing processing systems
2. Business-to-Consumer
   - “Good shopping experience” most important
   - Good, attractive user interfaces
   - Easy-to-use contextual help
   - Personalizations
   - Straightforward, efficient functionality

B. *Geography*

1. Language support
   - Translations in all areas of shopping experience or just some
   - Multilingual support of customer service

2. Currencies across countries

- Currency fluctuations
- Currency conversions
- Support for euro
- Pricing in local currency or U.S. dollars

3. Taxation by country

- Sales, value added tax, excise, national, local, duties, customs
- Possibility for tax exemptions

4. Shipping practices across countries

- Varying shipping charges by country
- Shipping range limitations

5. Products

- Different product lines by country
- Product price calculation and display by country

6. Customizing of merchandising strategy by country

- Use of color, images, and graphics by country
- Advertising perceptions by different cultures

7. Miscellaneous

- International privacy laws (what information can be requested of consumers by country)
- Encryption usage by country
- Single-byte versus double-byte coding

II. Store or Mall

- Logic of product line (all the same or widely different)
- Individualism (look and feel) of each store in a mall
- Different handling of processes for multiple stores required?

III. Competitors

- Global range of competitors
- Quality of service and products to differentiate yourself from others
- Online or offline competitors

IV. Shoppers

A. *Registered*

- Regular shopper
- Personal information retained in database
- Receive discounts and other promotional offers
- Personalization of shopping experience

B. *Guest*

- One-time-only shopper
- Personal information entered only for purchase, then deleted
- No promotional offers or personalizations available—promotional offers for all shoppers

V. Products

A. *Pricing*

- Sale prices and their expirations
- Discounts by quantity (or weight), customer, or category
- Inclusion of tax in product price issue

B. *Entire or subset of product line for online sale*

- Hierarchy by dividing products into categories
- Split very different categories to separate stores in a mall

VI. Payment Processing

- Cash or check acceptability
- SET, CyberCash, and other digital payment formats
- Credit card types accepted
- Credit and fraud checking
- Online or offline authorization
- authCapture or authOnly
- International payment conversions
- Plan to get SET certificates

VII. Order Processing

A. *Inventory*

- Inventory availability tracking
- Batch or real-time inventory updating
- Low inventory warnings to back-end system
- Limitation of order quantity amount
- Partial orders and back-order processes
- Shipping providers' shipping range

B. *Shipping*

- Possible internal or third-party fulfillment of orders
- Shipping cost calculation

  Net.Commerce delivers several possibilities, such as a flat shipping cost for a whole order, or shipping cost by quantity, weight, and so on for a special products or product groups. Or use a third-party package, legacy system calculation, or custom code.

- Bill-to and ship-to addresses
- Name and address verification

C. *Notification*

Customer confirmation of order only after inventory and shipping checked

D. *Order Status*

Order status checks at any time by customer

VIII. Security

- Encryption of transmissions
- Integrity of transmitted data
- SSL and SET protocols
- Digital certificate and key generation
- Firewall strategy
- Restriction of intranet and server access

- Authentication of users
- Physical location of hardware
- Monitoring and logging of system activity
- OS/400 security options
- OS/400 System Values—Set the OS/400 System security level to 50 with the command: `CHGSYSVAL SYSVAL(QSECURITY) VALUE(50)`

  50=Password, object, and enhanced operating system integrity

- Plan backup mechanisms

IX. Disclaimers and Store Policies

- Late deliveries
- Incorrect deliveries
- Out-of-stock products
- Returned products
- Canceled orders

X. Customer Service

- Full automation or partial automation
- Self-help by means of text files and frequently asked questions (FAQs)
- Customer feedback (e-mail or online forms)
- Opportunity to call customer service representatives on a support line

XI. Existing Methodologies

- Create interfaces to back-end systems
- EDI
- Other ERP

XII. Data Transfer

- Method of data transfer to an e-business database
- Correctness of data
- Possibility of manual data entry
- Method of data transfer from changes in your back-end system into your e-business database
- Method of data transfer from your e-business database (for example, new orders) to your back-end system

XIII. Performance

- Greatly affected by site traffic
- Hardware considerations
- Optimizations (user interface, database, code)
- Benchmarks to measure performance

XIV. Tools

- Reporting tools
- Administrative tools
- Custom tools

XV. Images and multimedia files

- Tools for creating these files
- Naming rules for images
- Using multimedia files carefully in case of performance

3.7 Output from the Design

This section describes which documents you should have created as a result of the requirements in the gathering and design phase.

3.7.1 Business Objectives

Build a list of functional requirements that answers the question: What should the application do? The summary checklist in 3.6, “Summary Checklist — Side Design Considerations” on page 41, can be helpful.

Group these requirements together into *functional units,* for example, *display product page.* Decide how each functional unit should be implemented. For example, we will integrate legacy system data with the application price calculation in Net.Commerce. Rank the requirements and objectives by priority and structure the project plan around this ranking system.

3.7.2 Navigation Flow

The navigation flow diagram is essentially a site map for the e-commerce site. It consists of the screens and how they flow. You need a diagram that shows the navigation flow for the entire e-business application and for all functional units in detail.

Figure 9 shows the navigation flow for our sample store. It is the fastest way to place an order in an e-business application. You can also see in which cases our back-end system is involved. For more information about our sample store, see 3.9, “Design of the ShopITSO Sample Solution” on page 57.

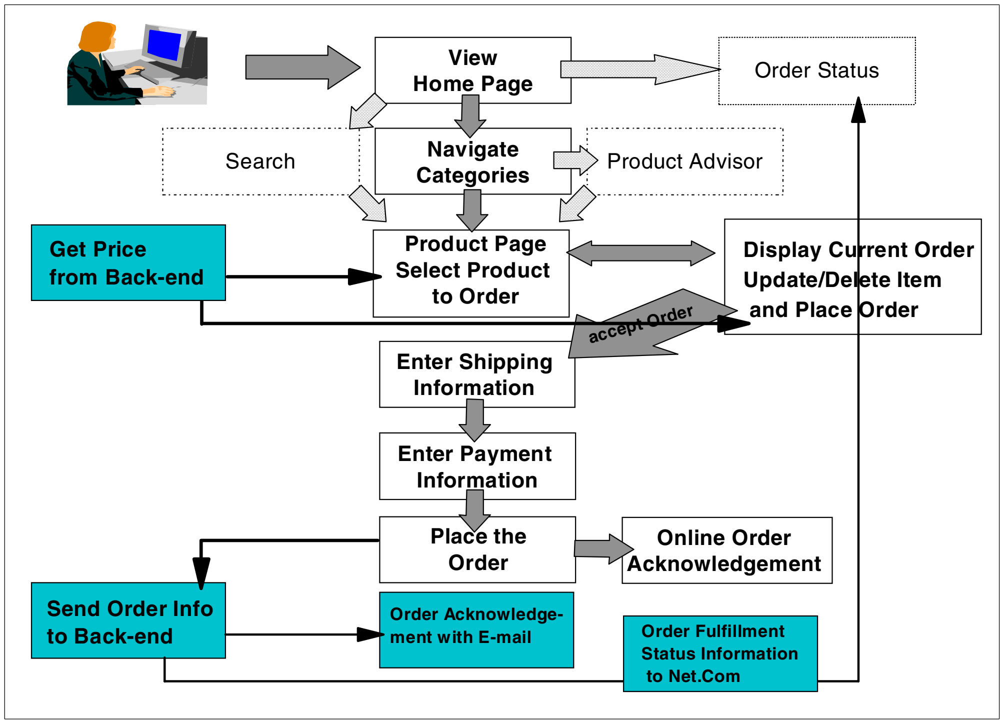

View Home Page
Order Status
Search
Navigate Categories
Product Advisor
Get Price from Back-end
Product Page Select Product to Order
Display Current Order Update/Delete Item and Place Order
accept Order
Enter Shipping Information
Enter Payment Information
Place the Order
Online Order Acknowledgement
Send Order Info to Back-end
Order Acknowledge-ment with E-mail
Order Fulfillment Status Information to Net.Com

Figure 9. Navigation Flow for the Entire e-Business Application

Note

There is one aspect that has to be handled with care. When you show the product page, you have no information about the quantity that the customer wants to order. At this point, you cannot deliver the discounted price for the product if you handle calculated discounts based on quantities. You have to decide which price information you want to show in this case. For example, show the list price and some text information about your special price rates.

In the DisplayCurrentOrder page, we do not have this problem. Here, we get the calculated price information from the back-end system (in our case) or from the Net.Commrce system. This can be any calculated price.

The navigation flow for the functional units shows all screens that are identified and are either unique screens or logical groupings of screens. A logical grouping of screens consist of screens that, except for the actual (dynamic) content of the page, have identical functionality in all ways. For example, a product page may be a logical grouping of screens if all the products in a given catalog are to be displayed identically. Figure 10 shows the flow for Display Product page from our sample solution.

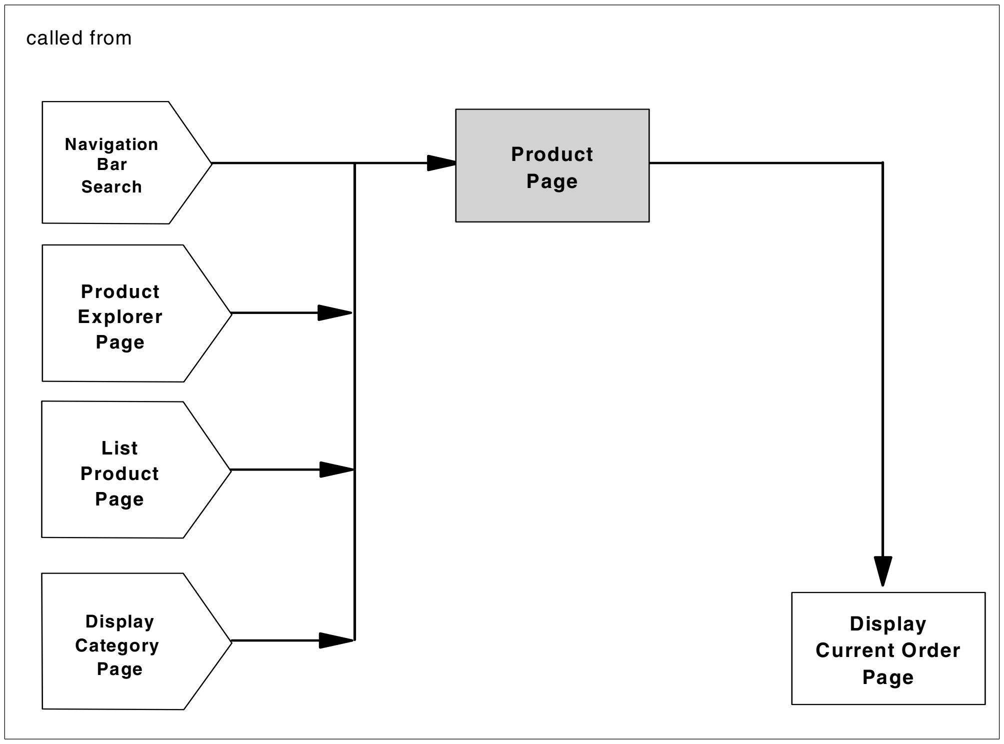
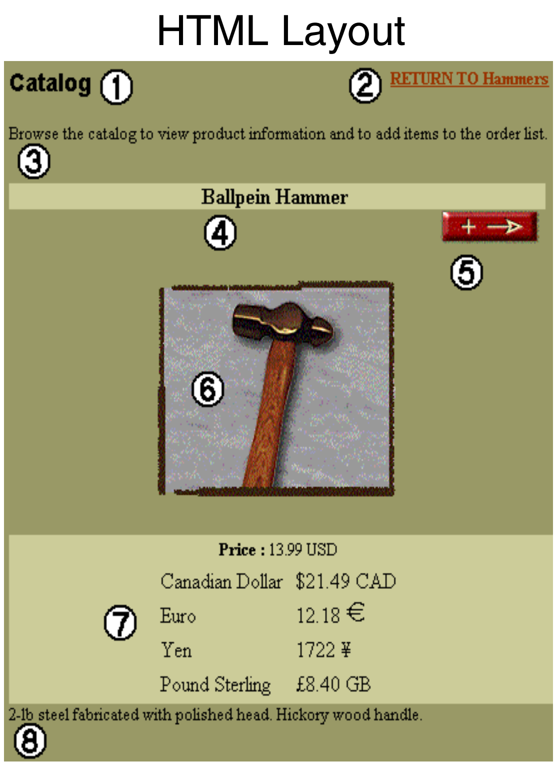


Figure 10. Sample Navigation Flow for Our Product Page

3.7.3 Functionality Description of Each Screen

For each unique screen in the site, you describe the screen content, what each screen does, and how it will work.

You also describe which images (product pictures, logos, or other multimedia files) have to be shown. You can see a sample of these specifications for a product page in Figure 11.


HTML Layout
Catalog
1
2
RETURN TO Hammers
Browse the catalog to view product information and to add items to the order list.
3
Ballpein Hammer
4
5
6
Price : 13.99 USD
Canadian Dollar $21.49 CAD
Euro 12.18 €
Yen 1722 ¥
Pound Sterling £8.40 GB
7
2-lb steel fabricated with polished head. Hickory wood handle.
8

Specifications

1. Static Text: "Catalog"
2. Link back to previous category template with static text "RETURN TO $VAR" where $VAR is the name of the parent category. $VAR is generated from the name of the parent category and will not exceed 20 characters.
3. Static text with instructions on what this page does. Text will not exceed 200 characters, but may wrap into several lines on the screen.
4. Dynamic text generated from the database with the name of the product displayed. Text length will not exceed 35 characters.
5. Link to add this product to the order. The image for the link is static.
6. Dynamic picture of the product being displayed.
7. Multilined text that shows the product price in the various currencies supported. One currency per line.
8. Dynamic description of the product. This is a multilined text field generated from the database. The text may be as long as 2000 characters and should flow down the page.

Figure 11. Sample of Functionally Description of Display Product Page

3.8 Mapping Your Navigation Flow to the Net.Commerce Commands

The shopping process is the flow of information from when the shopper enters the store to when an order has been processed. Each step within the shopping process either adds data to the Net.Commerce database or uses data in the database to display a page. The default shopping process uses a specific group of Net.Commerce commands in a defined order.

There are many ways that this process can be customized, both in the information that is displayed at each stage and the number of steps actually required. Commands, tasks, and overridable functions (OFs) represent the basic building blocks of the Net.Commerce system to give this flexibility. All of the navigation in the Net.Commerce e-business application is done by using these commands. Together with the database these commands are the heart of Net.Commerce.

Before you map your navigation flow for your e-business application to the Net.Commerce commands, you have to know what these commands are doing in the shopping process. You should read the handbook *Net.Commerce for AS/400 Commands, Tasks, Overridable Functions, and Database Tables* (dbtodcmd.pdf), which comes with the Net.Commerce product.

Note

You can find the handbook *Net.Commerce for AS/400 Commands, Tasks, Overridable Functions, and Database Tables* (dbtodcmd.pdf), in the directory: /QIBM/ProdData/NetCommerce/html/MRInnnn/ncbooks

The nnnn is your language code, for example, 2924 for the English language.

The implementation of the shopping process follows the sequence described in the following text.

Net.Data is used as the main display engine for the Net.Commerce system. As the display engine, Net.Data should not be used to implement business logic.

To display information in a browser screen, you use a Net.Data macro that provides the SQL statements (select) to get the required data from the database. The resulting data from the SQL statement is converted into the HTML format (a Net.Data function). This data is then shown by the browser. One example of this is to show the Current Order page.

Implementing business logic is best left to commands and overridable functions (OFs), instead of Net.Data. Commands and OFs are the recommended engine for database inserts or updates. Net.Data should be used as a *display tool* only.

The Net.Data macro also uses (calls) the Net.Commerce commands that are responsible for the shopping process and the UPDATE of the database tables. For more information, see 3.8.2, “Using Net.Commerce Commands” on page 52.

Your next step is to decide which Net.Commerce commands fit the requirements in your navigation flow. The following sections describe how you can do this.

Important

Do not use the SQL INSERT, UPDATE and DELETE statements in the Net.Data macros to manipulate the database. Although Net.Data supports these SQL statements, we *strongly recommend* that these never be used in any macro in a Net.Commerce system.

Use the Net.Commerce commands for this. The Net.Commerce commands are using *commit* and *rollback* functions and guarantee the integrity of your database.

3.8.1 Overview of Net.Commerce Commands

A Net.Commerce command represents a static business process that delegates well-defined pieces of business logic to tasks. Net.Commerce commands call tasks, which are mapped to OFs. Indirectly, Net.Commerce commands call OFs through tasks. They can also indirectly call other commands, and OFs can call other tasks. See Figure 12 on page 49.

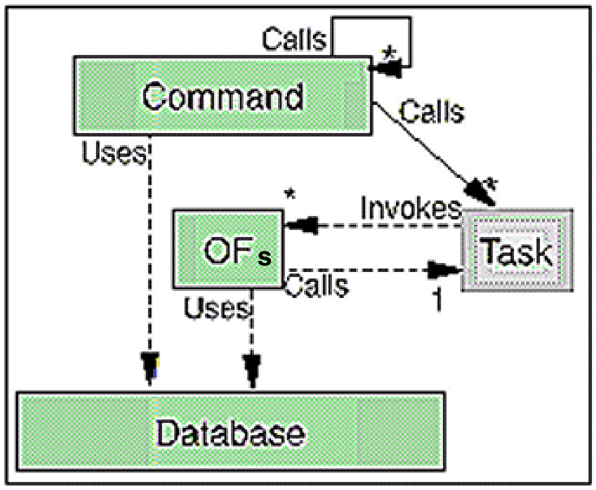


Figure 12. Commands, Tasks, and Overridable Functions

A command is hard coded to call a set of one or more tasks. The command may call a different task from the set based on the input to the command. This means that different OFs will be called based on the input to the command. A task may be mapped to OFs at the store level. This makes it possible to have a task with the same name in different stores execute different logic.

Commands and OFs are actual pieces of code, but tasks are not. Tasks are more like specifications. They are a contract that defines the behavior and the input and output parameters by which a called OF must abide for the system to function properly. The Net.Commerce commands can also handle error situations.

Commands work together with command parameters. Most commands have required parameters and optional parameters. For an example, see Figure 15 on page 52. The most important Net.Commerce commands are listed in Figure 13 on page 50.

For detailed information about commands, tasks, and OFs, see the handbook *Commands, Tasks, Overridable Functions, and Database Tables*, which comes with the Net.Commerce product. You can also refer to Chapter 19, “Implementing Overridable Functions” on page 413, in this redbook.

Purpose	VIEW commands	PROCESS commands	Format#
Shopper Logon	LogonForm	Logon	LogonForm?url=*u#* Logon#
View Category Page	CategoryDisplay		CategoryDisplay?cgmenbr=*m*&cgrfnbr=*c*
View Product Page	ProductDisplay		ProductDisplay?prmenbr=*m*&prrfnbr=*p*
Shopper Registration	RegisterForm	RegisterNew RegisterUpdate	RegisterForm# Register
Shopper Address Book	AddressForm	AddressAdd AddressUpdate AddressDelete AddressCheck	AddressAdd# AddressUpdate# AddressDelete# AddressCheck#
Manage Shopping List	InterestItem Display	InterestItemAdd InterestItemDelete	InterestItemDisplay?merchant_rn=*m* InterestItemAdd# InterestItemDelete#
Manage Sub-orders	OrderItemList OrderItemDisplay	OrderItemProcess OrderItemUpdate OrderShippingUpdate	OrderItemProcess OrderItemUpdate OrderShippingUpdate OrderItemDisplay
Manage Orders	OrderList OrderDisplay	OrderProcess OrderUnlock OrderCancel	OrderList?status=*P** OrderDisplay?status=*P**
General Purpose	ExecMacro ExecTask	ExecUrl	ExecMacro/macro-name.d2w/report?parameters#

u = URL of next page, *m* = Merchant Ref Number, *c* = Category Ref Number, *p* = Product Ref Number

* order status: *P* = pending Order, *c* = completed Order

# *full syntax for command parameter, see handbook:* Commands, Tasks, Overridable Functions, and Database Tables, which comes with the Net.Commerce product

Figure 13. Important Net.Commerce Commands

Net.Commerce V3 includes a new flexible server architecture that allows new functions and commands to be added to the server by registering them in the database. When a command is sent to the server, it verifies that the command is valid by checking it against the database and invokes the main command function, which is a C++ function.

There are two different types of Net.Commerce commands:

- **VIEW** or **user-interface (UI) commands** — Retrieve information from the Net.Commerce database and display store pages. This function is achieved by calling Net.Data macros that combine SQL statements with HTML.
- **PROCESS** or **non-user-interface (non-UI) commands** — Process and write information to the Net.Commerce database. These commands are typically invoked by submitting input forms during the shopping process.

Many commands offer some customization ability by calling tasks, which are registered in the database. Each task has an overridable function assigned that you can override with your own function (either for the site or per store) to change the way it processes information. You can also turn off some tasks (for example, inventory checking) by assigning the overridable function `DoNothingNoArgs` to the task.

VIEW commands always finish processing by invoking a VIEW task. This VIEW task usually has the overridable function `TaskDisplay` assigned, which uses Net.Data to run a macro. The name of the macro is determined from the database. Refer to 13.15.2, “Finding or Assigning a Net.Data Macro for a Specific Display” on page 244, for more information.

Most Net.Commerce commands work together with command parameters. Most commands have required parameters and optional parameters. For an example, see Figure 15 on page 52. Figure 14 shows an overview of the command processing.

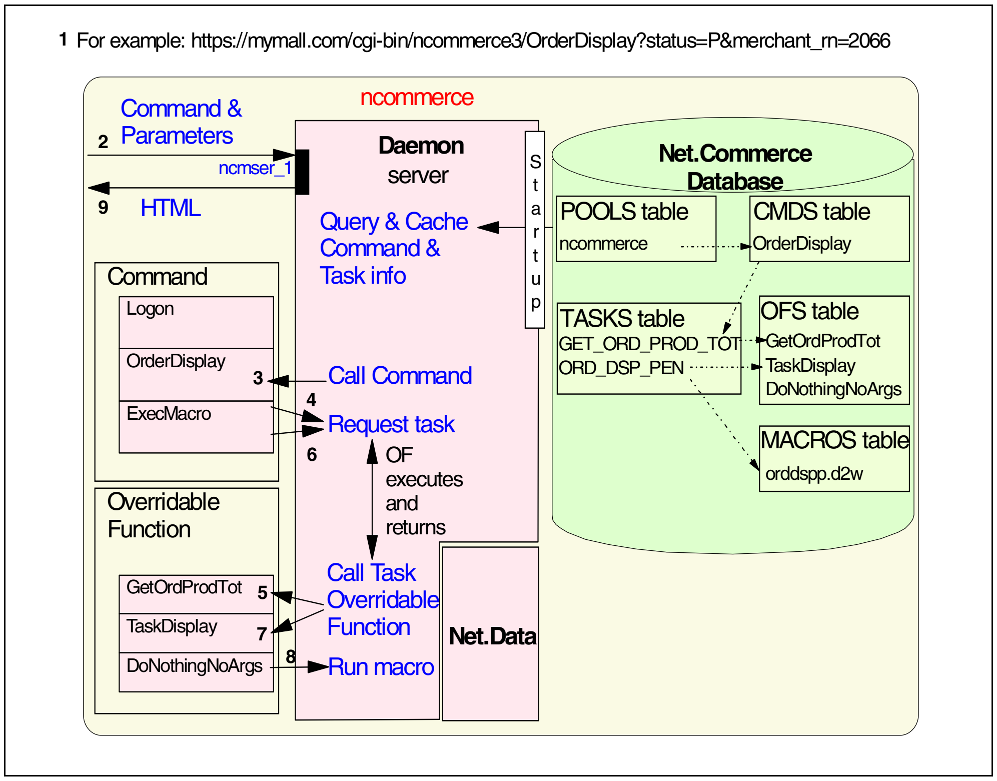


Figure 14. Command Processing

The server loads the information about pools, commands, and tasks into a memory cache when it starts (startup). This allows command requests to be validated and processed quickly.

When a URL enters the system **(1)**, the processing is started. Based on the URL, the HTTP server passes control to the Net.Commerce director **(2)**.

When a command is sent to the server, it verifies that the command is valid and invokes the main command function **(3)**, which is a C++ function. The command requests that an individual task be executed based upon the parameters **(4) (6)**. Each task calls the overridable function (OF) mapped to the task **(5) (7)**. When an OF completes, it returns control to the command and the next task is called. In this example, we are looking at a VIEW type command. For this reason, the last task will cause a Net.Data macro to be executed **(8)**. The Net.Data macro builds HTML output and returns it the HTTP Web server **(9)**.

Many commands offer some customization ability by calling one or more tasks. The server is asked to run the task. It does this by calling the overridable function assigned for that merchant (or for the mall if there is no merchant assignment).

Overridable functions are also C++ functions. VIEW tasks and ERROR tasks usually have the overridable function TaskDisplay assigned, which uses Net.Data to run a macro. The name of the macro is determined from the MACROS table.

3.8.2 Using Net.Commerce Commands

Figure 15 shows the syntax of the Net.Commerce commands and the resulting URL that is used from an HTML document or Net.Data macro to execute the command. The */cgi-bin/ncommerce3/* portion of the URL causes the HTTP server to execute the Net.Commerce Director program. The Director passes the command name with the parameters to the Net.Commerce Daemon. In this case, the command is *OrderList*.

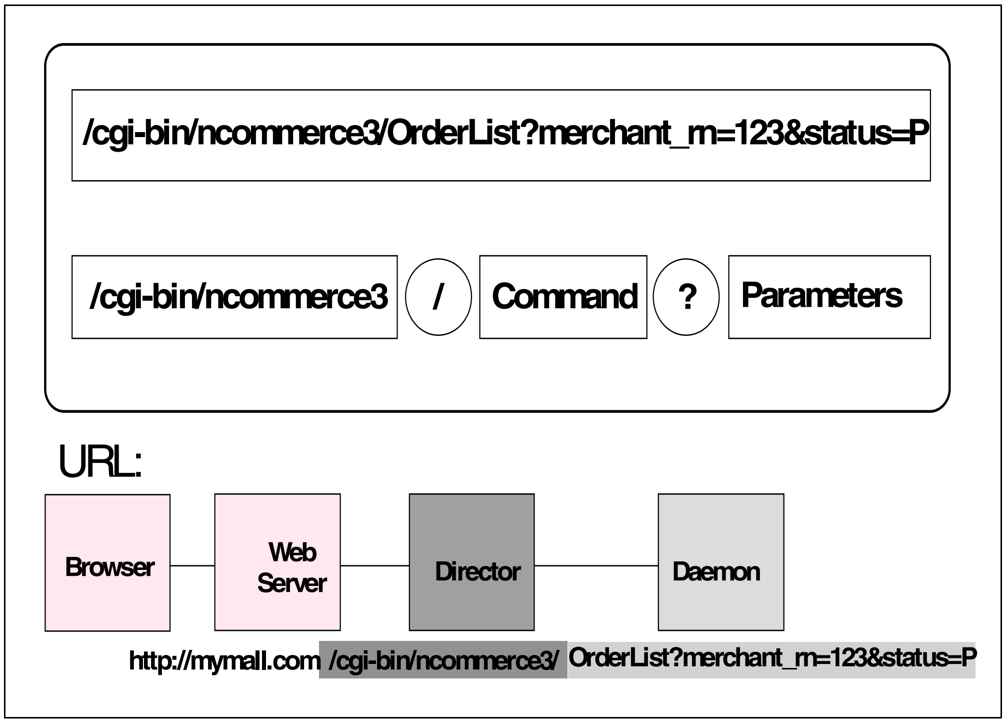


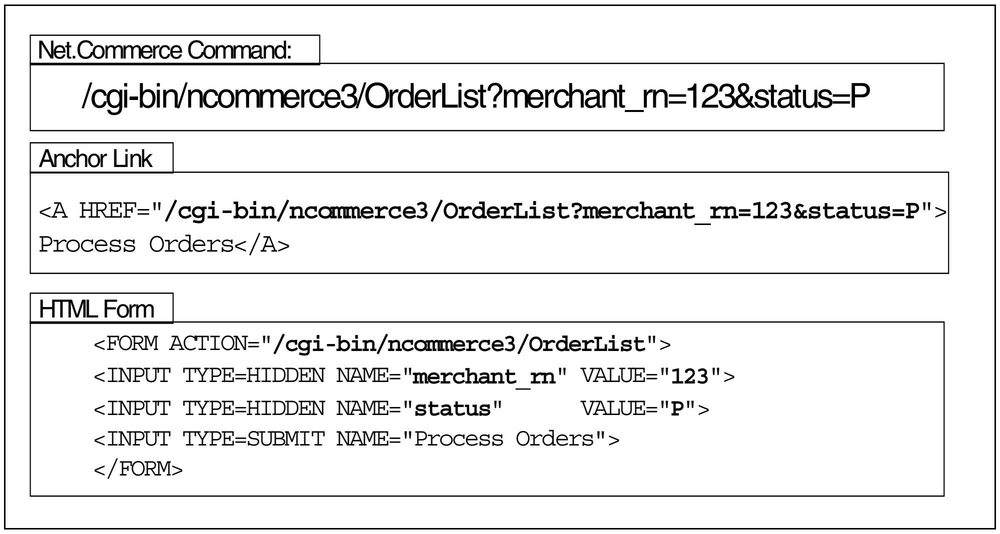
Figure 15. Sample of Syntax of Net.Commerce Commands

Important

Be sure to include the question mark. It is part of the Net.Commerce command syntax. It separates the command name from the parameters.

3.8.2.1 Using Net.Commerce Commands in HTML

To use the Net.Commerce commands in your static HTML or Net.Data generated HTML, you must construct the command and the correct parameters. Figure 16 shows two techniques for doing this.



```
/cgi-bin/ncommerce3/OrderList?merchant_rn=123&status=P
```

```
<A HREF="/cgi-bin/ncommerce3/OrderList?merchant_rn=123&status=P">
Process Orders</A>
```

```
    <FORM ACTION="/cgi-bin/ncommerce3/OrderList">
    <INPUT TYPE=HIDDEN NAME="merchant_rn" VALUE="123">
    <INPUT TYPE=HIDDEN NAME="status"      VALUE="P">
    <INPUT TYPE=SUBMIT NAME="Process Orders">
    </FORM>
```

Figure 16. How to Use Commands in HTML or Net.Data Macros

The first way is to construct an anchor link in the HTML. The HREF value contains all three parts needed to cause the command to be executed. The second way shown is through the use of an HTML form. The action contains the path and the command, while the parameters come from hidden input fields. The GET method is used so that all the information is put together in the URL that is sent to the HTTP server.

Note

When you type a command in a macro, use the following conventions:

- Ensure that there are no spaces between keywords, delimiter, and variables.
- Enter parameters in any order.
- Enter keywords in lower case.
- Enter variables that represent column names and table names in lowercase.
- Replace spaces in values with plus signs (+).

When entering a command in a hypertext link or as the value of a URL parameter (redirection URL), replace any symbols with their hex values. This does not apply to commands contained in the URL of form submit buttons. The following is a list of some common substitutions:

- Replace the slash (/) with `%2F`
- Replace the question mark (?) with `%3F`
- Replace the ampersand (&) with `%3D`
- Replace spaces in values with plus signs (+)

3.8.3 Mapping the Navigation Flow to Net.Commerce Commands

Your next step in the design of your e-business solution is to search for the Net.Commerce commands that fit your navigation flow and the behavior of this flow. For our sample store, we did this for all the Web pages (functional units). Figure 17 shows an example of our navigation flow mapping to the Net.Commerce commands for our Product page.

We searched for a Net.Commerce command that handles orders in the commands table (Figure 13 on page 50). We found the OrderItemUpdate command.

Using the *Commands, Tasks, Overridable Functions and Database Tables* handbook (included with the Net.Commerce product), we now verified that this command is the one we want to use. The behavior, syntax and parameters are described in detail in this handbook. For more information, see 3.8.3.1, “Description of Net.Commerce Command OrderItemUpdate” on page 54.

Another way to find the Net.Commerce commands is to look at the Net.Data macros that are built when you use the Store Creator to build a new store. For more information about the Store Creator, refer to 13.3, “Building the Store with Store Creator” on page 187.

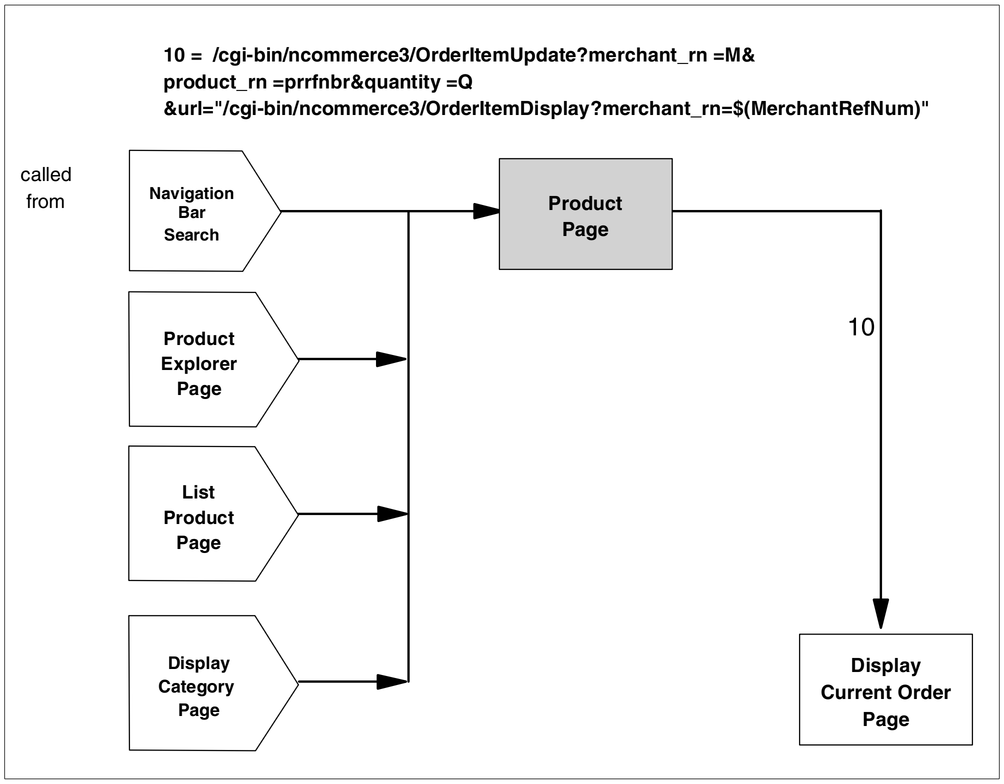

10 = /cgi-bin/ncommerce3/OrderItemUpdate?merchant_rn =M&
product_rn =prrfnbr&quantity =Q
&url="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)"
called from
Navigation Bar Search
Product Explorer Page
List Product Page
Display Category Page
Product Page
10
Display Current Order Page

Figure 17. Navigation Flow to Net.Commerce Command Mapping

3.8.3.1 Description of Net.Commerce Command OrderItemUpdate

This section is an excerpt from the *Commands, Task, Overridable Functions, and Database Tables* handbook that illustrates the command syntax of a Net.Commerce command.

The OrderItemUpdate command updates or creates a shipping record, depending on whether the shipto_rn or product_rn is passed to it, respectively.

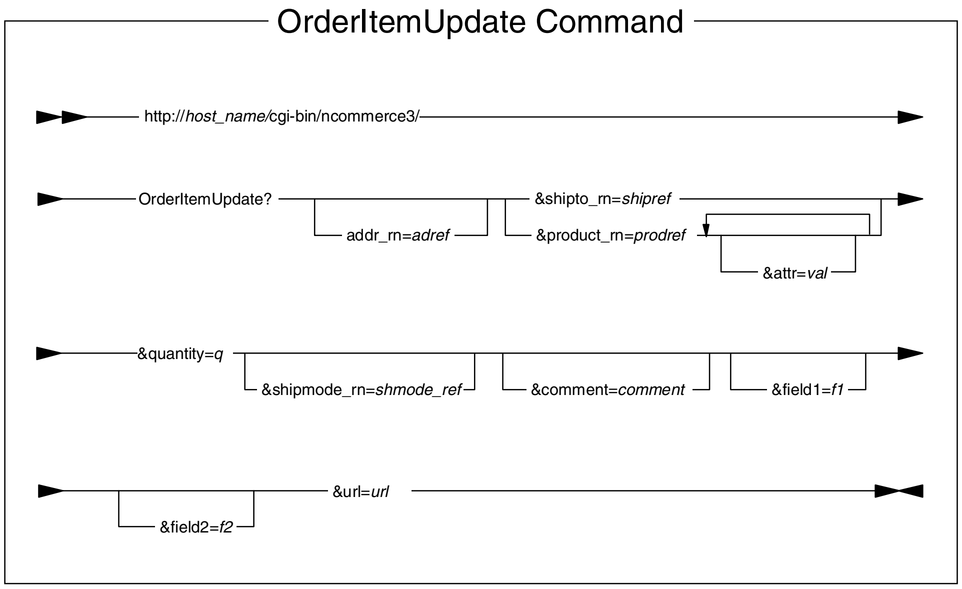


Figure 18. OrderItemUpdate Command Syntax Diagram

Parameter Values

The parameter values shown in Figure 18 are explained in the following list:

- **host_name**

  The fully qualified name of the Net.Commerce server.

- **addref**

  The reference number of the address to which the products and items are to be shipped. This is an optional parameter.

- **shipref**

  The reference number of the shipping association.

- **prodref**

  The reference number of the product whose attributes are to be updated.

- **attr**

  Any distinct attribute that is defined for the product in the table PRODDSTATR.

- **val**

  The value of the distinct attribute.

- **q**

  The quantity of the product or item to be shipped.

- **shipmoderef**

  The reference number of the shipping mode to be used for the product or item.

- **comment**

  A comment to be included with the order.

- **f1**

  A merchant-reserved integer value.
- **f2**

  A merchant-reserved text value. Accepts up to 254 characters.
- **url**

  The URL that is called when the command successfully completes.

Behavior

The addr_rn parameter is optional. If it is not supplied, the parameter defaults to the address reference number of the row in table SHADDR, where column SAADRFLG has the value "P" and the value in column SANICK is the shopper's ID from column SHLOGID in the SHOPPER table. If there is no such row in the table SHADDR, then it creates a new row with all the other columns being NULL.

If shipto_rn is specified, it updates the entry with the specified quantity. If product_rn is specified, it determines the SKU number based on the specified attributes, and creates a shipto entry for the specified quantity of the item.

The command checks the validity of the address book reference number and updates it if necessary. The number is incorrect if it refers to an address book entry that the shopper changed after creating the shipping association. Then, for both shipto_rn and product_rn, this command performs the following steps:

1. Unlocks the order if it is locked.
2. Calls the CHECK_INV process task to ensure there is enough inventory.
3. Calls the GET_CURRENCY process task to determine the currency to be used.
4. Calls the GET_BASE_SPE_PRC process task to get the special price associated with the product or item.
5. Stores the output parameters BASE_CURRENCY and BASE_PRODUCT_PRICE from the GET_BASE_SPE_PRC process task in the STBASEPRICE and STBASECURR columns in the SHIPTO table.
6. Calls the RESOLVE_SKU process task to determine the SKU for each product or item.
7. Updates the comment, field1, and field2 fields.
8. Calls the EXT_SHIPTO_UPD process task to perform additional processing to meet any unique requirements.

On successful completion, the command calls the URL specified. It also rounds monetary amounts before storing the information in the database, according to the rounding information defined in the CURRFORMAT and SETCURR tables.

Exception Conditions

Note the following conditions:

- If the overridable function assigned to the RESOLVE_SKU process task determines that a required product attribute is missing, it sets the BAD_PROD_ATTR exception task to handle the error.

- If it determines that a product with the specified attributes does not exist in the database, it sets the MISSING_PROD_ATTR exception task to handle the error.
- If the quantity specified is not numeric or not a positive value or errors are detected in the values for comment, f1, or f2, it sets the BAD_ST_DATA exception task to handle the error.
- If the overridable function assigned to the CHECK_INV process task fails, it sets the CHECK_INV_ERR exception task to handle the error.
- If the overridable function assigned to the GET_BASE_SPE_PRC process task fails, it sets the GET_BASE_SPE_PRC_ERR exception task to handle the error.
- If the overridable function assigned to the EXT_SHIPTO_UPD process task fails, it sets the EXT_SHIPTO_PROC_ERR exception task to handle the error.

3.9 Design of the ShopITSO Sample Solution

This section shows you the output of our design phase, which builds a description of our sample store ShopITSO.

3.9.1 Business Objectives in the ShopITSO Sample Store

Following the Side Design Considerations Checklist as described in 3.6, "Summary Checklist — Side Design Considerations" on page 41, our sample store is based on the following business objectives:

I. Audience and Scope

A. *Business Type*

Our store sells computers to consumers in the USA only.

Good, attractive user interfaces that are easy to use and come quickly to the page where the order can be placed are most important.

As you can see from our navigation flow, we do not use the shopping cart function that Net.Commerce delivers. Our customers place selected products directly to the current order.

B. *Geography*

1. Our store is built in just one language.
2. Pricing is in local currency—U.S. dollars.
3. We have a fixed tax rate for all of our products in the entire mall.
4. We will not ship outside the USA.
5. We sell one product line for the USA.
6. Merchandising is done from our home page, which includes promotions for products.
7. Miscellaneous—There are no special considerations for our store.

II. Store or Mall

Our business is done by using one store (so in our case it is also a mall).

III. Competitors

There are no special considerations for our store.

IV. Shoppers

We decided to sell our product to one-time-only shoppers.

Customer registration is not necessary in our store.

V. Products

A. *Pricing*

We use our back-end system to get the sale price for the products.

B. We sell a *subset of products* in our online store.

We build a hierarchy by dividing products into categories.

VI. Payment Processing

We offer our customers two methods of payment:

1. We accept credit cards from VISA without using SET.
2. We offer our customers the use of SET for secure payment through credit cards.

We also use the *Merchant Originated Payment* option (see 6.2.1, "Merchant Originated Payment" on page 96).

Our back-end system signals when an order has been delivered. The batch process for request capture by the acquirer is then done automatically from the payment gateway. We did not use the Store Manager function for request capture).

VI. Order Processing

A. *Inventory*

The back-end system gets information of all placed orders from the e-business application. Order fulfillment and inventory update is made in the back-end system.

We allow back order. We have no inventory check in our e-business system.

B. *Shipping*

We have three shipping practices that are valid for the entire order, each of them with a special amount of shipping cost for the customer:

- Delivery in 24 hours
- Delivery in 48 hours
- Delivery in 72 hours

We use a flat shipping cost calculation that is done by Net.Commerce (see 13.3, "Building the Store with Store Creator" on page 187).

The bill-to and ship-to addresses are the same. The address is entered by the customer before submitting the order.

Name and address verification is the default implementation of Net.Commerce (changes are possible through a new or changed overridable function).

C. *Notification*

Online customer confirmation that the order has been received is done using Net.Commerce after online order validation is complete (address is present and payment information or payment request authorization is complete).

Order acknowledgement is from the back-end system through e-mail.

D. *Order Status*

Order status checks by the customer are possible at anytime. The back-end system informs Net.Commerce as the order status changes (for example, order shipped).

VII. Security

We use SSL protocol and digital certificates. In our test system, the certificates are generated by the AS/400 server, so we did not use certificates from a certification authority.

For the payment function, we use the SET protocol with SET certificates.

The firewall, Net.Commerce and HTTP Web server are all on the same AS/400 system. The back-end system is another AS/400 system.

We work with OS/400 system security level 50 (system value).

VIII. Disclaimers and Store Policies

These will be part of our home page through a link.

IX. Customer Service

Self-help by means of text files and frequently asked questions (FAQs) is planned and possible for our ShopITSO. Implementation will be in the first release using Lotus Notes.

Customers will also be given an opportunity to call customer service representatives on a support line.

X. Existing Methodologies

Our back-end system is integrated. It delivers the selling price for the product. It also keeps the Net.Commerce system updated with the changes in the order status. The back-end system gets all the orders placed from Net.Commerce for further processing.

Therefore, some integration efforts in modifying existing application programs should be calculated and planned.

See Chapter 5, “Planning: Integration with the Back-End Systems” on page 87, for more information.

XI. Data Transfer

For the method of data transfer to e-business database, see Chapter 15, “Importing Business Data into Net.Commerce” on page 319.

For the method of data transfer from Net.Commerce to back-end system, see Chapter 17, “Interfacing to Our Back-End Business System” on page 383.

XII. Performance

We use the caching options from Net.Commerce for the category and product pages.

Also, our back-end system uses triggers to update the Net.Commerce data when changes are made to the products in the back-end system. This takes care of the caching mechanism, when the price for a product is changed.

For the static HTML pages and the GIF files, we use the AS/400 HTTP server local cache function.

We also made changes in the Net.Data macros to reduce the amount of SQL queries to the database.

XIII.Tools

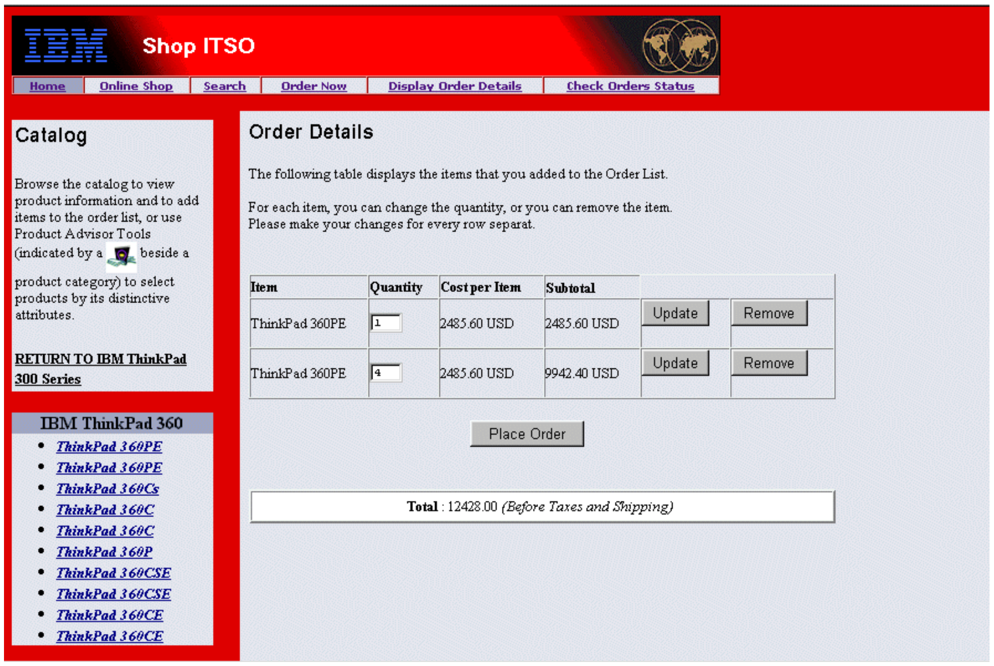
Refer to Chapter 7, “Planning: Tools to Build the Site” on page 103.

3.9.2 Functional Description of Each Page in the ShopITSO Sample Store

For all of our store pages, we described in detail which elements are on the page and the behavior of the page. In this section, we describe only one sample of our pages—the Display Current Order (Display Order Details in this implementation) page shown in Figure 19.

The navigation bar (top frame) and catalog tree (left frame) is described in 3.9.4.1, “Navigation Flow—Navigation Bar in Frames” on page 62. The content for pages shown in the main (right) frame are described in 3.9.4, “Navigation Flow and Net.Commerce Commands in ShopITSO” on page 62. These pages are Search, Order Now, Display Order Details (Display Current Order), Check Order Status, Product page, and all pages that follow after a product is selected for buying. The content of each page is described in detail.

Figure 19 shows you a sample so that you can see the kind of documentation that you may want to generate when designing your site.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

IBM ThinkPad 360
- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

Order Details

The following table displays the items that you added to the Order List.

For each item, you can change the quantity, or you can remove the item.
Please make your changes for every row separat.

Item	Quantity	Cost per Item	Subtotal		
ThinkPad 360PE	1	2485.60 USD	2485.60 USD	Update	Remove
ThinkPad 360PE	4	2485.60 USD	9942.40 USD	Update	Remove

Place Order

Total : 12428.00 *(Before Taxes and Shipping)*

Figure 19. Our Display Current Order Page in ShopITSO

To build this window, we use the shipto.d2w macro. The first text that you see in the Current Order page (in the right frame) is static HTML text. Then, you see a table with the product description, quantity ordered, cost per item, subtotal cost, and buttons for update and delete. This table is built through the SQL query to the SHIPTO table, which contains the item rows for all orders. The subtotal value for each row is calculated (it is not an element of the SHIPTO table). The total value of the order is also calculated.

The quantity field is an HTML form input field that allows the customer to change the quantity for any product listed. The Update button is used to update the SHIPTO table quantity with the value typed in this quantity field (one per row). The Remove button allows the customer to delete one product. In both cases, the next window that is shown is the recalculated Display Current Order page (the same page with the new values) or the NONE Order page, if there are no more items in the order list for the customer (all products have been removed).

With the Place Order button, the customer places the order and the next page (the Order Accepted page) will be shown.

3.9.3 Navigation Flow in the ShopITSO Sample Store

Review our total navigation flow of our sample store (Figure 20 on page 62). For a quick navigation through the store, we do not want to use a shopping cart (interested items). In our solution, the customer places products directly into a pending order (select products to order). As long as the customer does not close the browser, this pending order is available (Display Current Order) and can be changed by the customer. The customer can change the quantity they want to order and can also remove a product form the order list. Next, they place the order. Then, they must enter the shipping and billing address (both addresses are the same in our solution) and the payment information. We offer two methods to pay through credit cards, with or without the SET protocol.

Notice also that our back-end system is integrated. We use our back-end application programs to get the price for a product. This can be any calculation that the application program delivers.

To show the price in the product page, we get the list price from the back-end system. For the Display Current Order page, we get the discounted price from the back-end system.

When the customer places their order, the back-end system receives the information from the Net.Commerce application about the order data. The back-end system does all the order fulfillment. Also, the back-end system sends information about the status of the order (for example, order shipped) to the Net.Commerce system (change the status of the order). This allows the customer to see this information in the Order Status page.

Read more information about back-end system integration in Chapter 17, “Interfacing to Our Back-End Business System” on page 383.

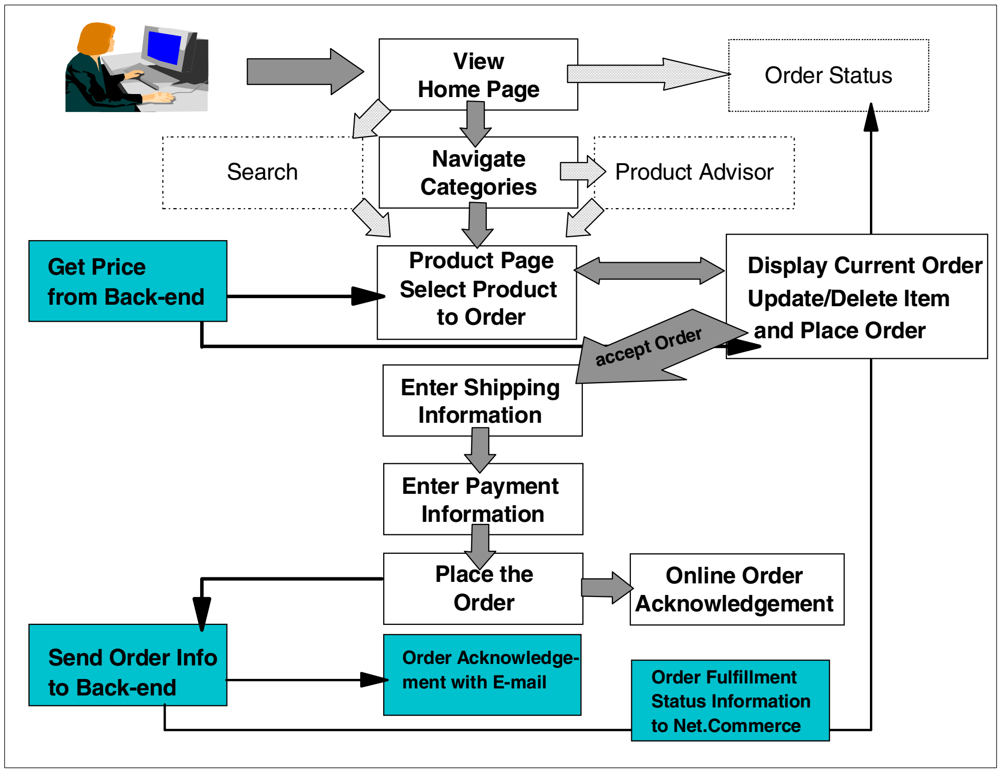

View Home Page
Order Status
Search
Navigate Categories
Product Advisor
Get Price from Back-end
Product Page Select Product to Order
Display Current Order Update/Delete Item and Place Order
accept Order
Enter Shipping Information
Enter Payment Information
Place the Order
Online Order Acknowledgement
Send Order Info to Back-end
Order Acknowledge-ment with E-mail
Order Fulfillment Status Information to Net.Commerce

Figure 20. Overview of Our e-Business Sample Application

To reduce the amount of pages in this redbook, we decided to show our navigation flow for each page together with the Net.Commerce commands, which we use in the following section.

3.9.4 Navigation Flow and Net.Commerce Commands in ShopITSO

The figures in the following sections show the navigation flow of all our pages that we want to show to the customer in our application sample store. You can see from which page or pages a specific page is called, and which page is shown, after the customer has made their choices.

3.9.4.1 Navigation Flow—Navigation Bar in Frames

The first page that we present to our customer is the home page with welcome text. This home page has two frames. The first frame (the top frame) has the banner image with the navigation bar,1 which has six buttons as shown in Figure 21 on page 63. The second frame (the main frame) shows the HTML home page shown in Figure 22 on page 63.

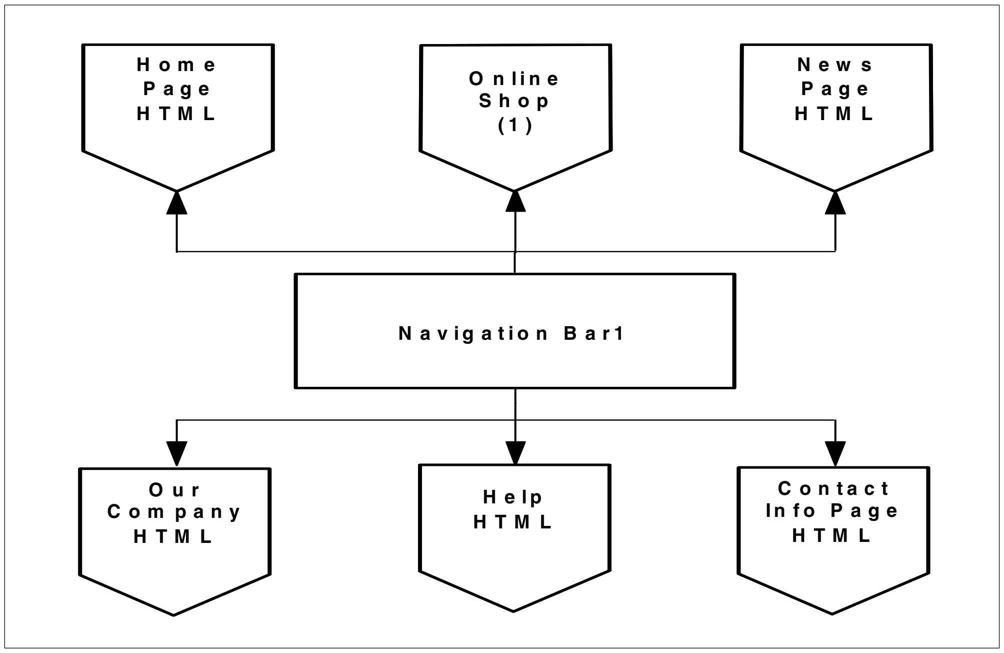
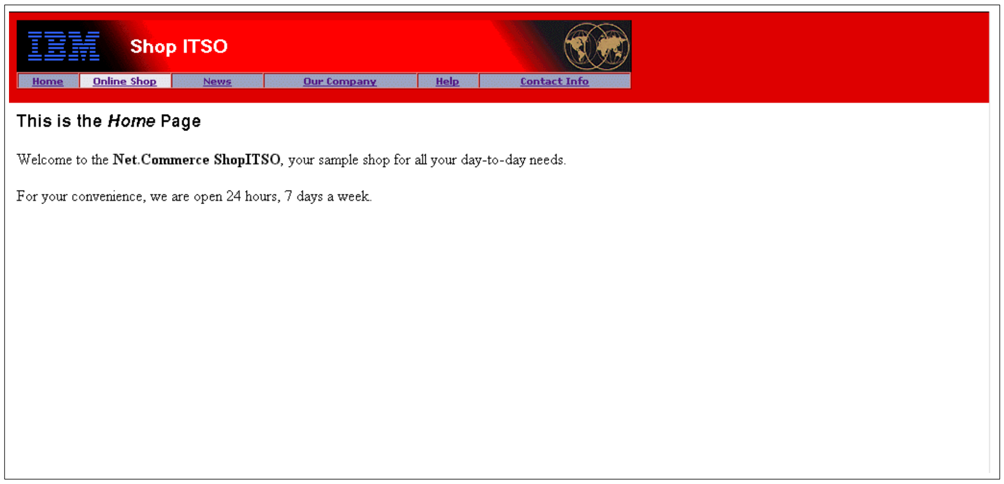


Figure 21. Navigation Bar 1

All pages that are reachable from the navigation bar1 are HTML pages, with the exception of the Online Shop. These HTML pages are shown in the second frame (the main frame) through an HTML link from the navigation bar1. All the HTML pages work with the same frames, the top and main frames, so they also all have the same navigation bar.



This is the *Home* Page

Welcome to the **Net.Commerce ShopITSO**, your sample shop for all your day-to-day needs.

For your convenience, we are open 24 hours, 7 days a week.

Figure 22. Home Page with Navigation Bar 1

When the customer clicks the Online Shop button from navigation bar1, the first page of our Net.Commerce e-business application, which is the Category Tree in the left frame of a new window, is shown. All pages in our Net.Commerce e-business application work with three frames. The top frame with the image and navigation bar2, the left frame with the catalog tree, and the main frame where all other pages such as the Product page are shown in Figure 23 on page 64.

You can see navigation bar2 which is used for all pages in our shopping process.

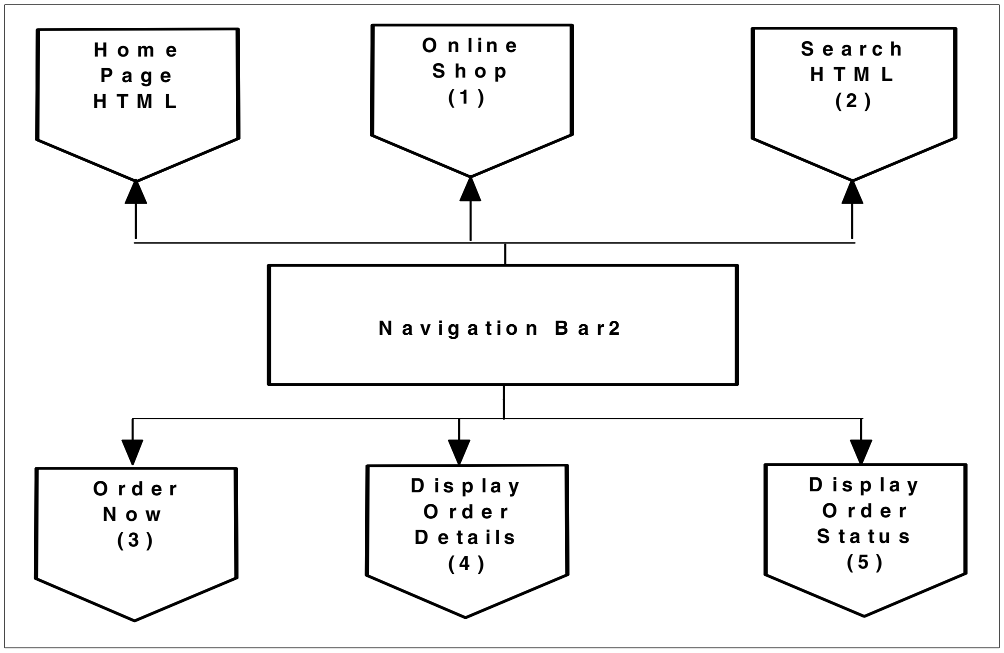


Figure 23. Navigation Bar 2 — Active when StartShopping Selected

3.9.4.2 Navigation Flow Home Page

Figure 24 shows the navigation possibilities, starting from our home page. The links are done through our navigation bar1.

All pages that are marked with HTML (News, Our Company, Help, Contact Info, and Promotion) are shown in the second frame (the main frame).

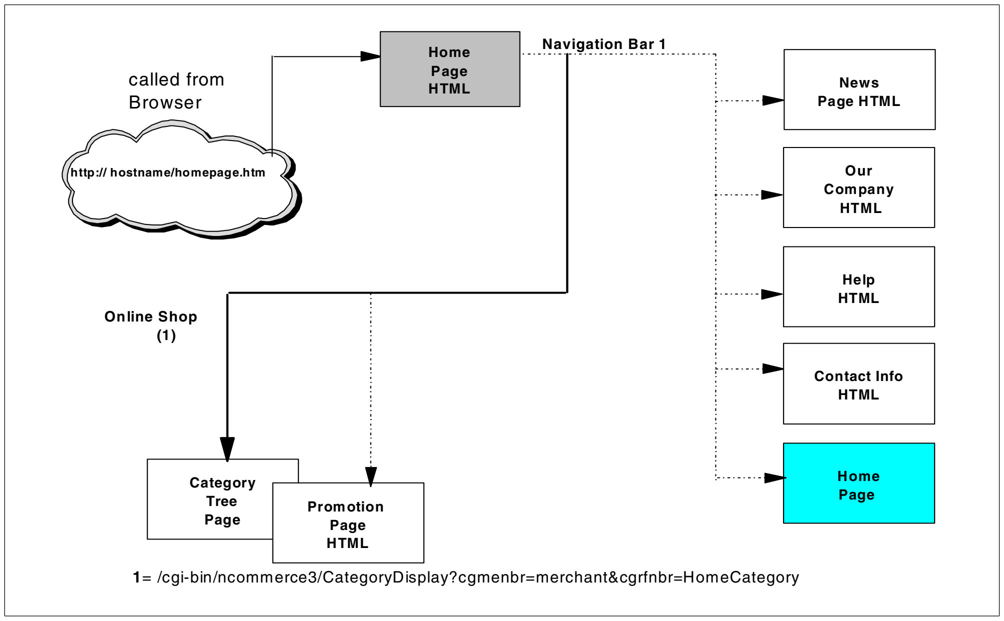


Figure 24. Navigation Flow from Our Home Page

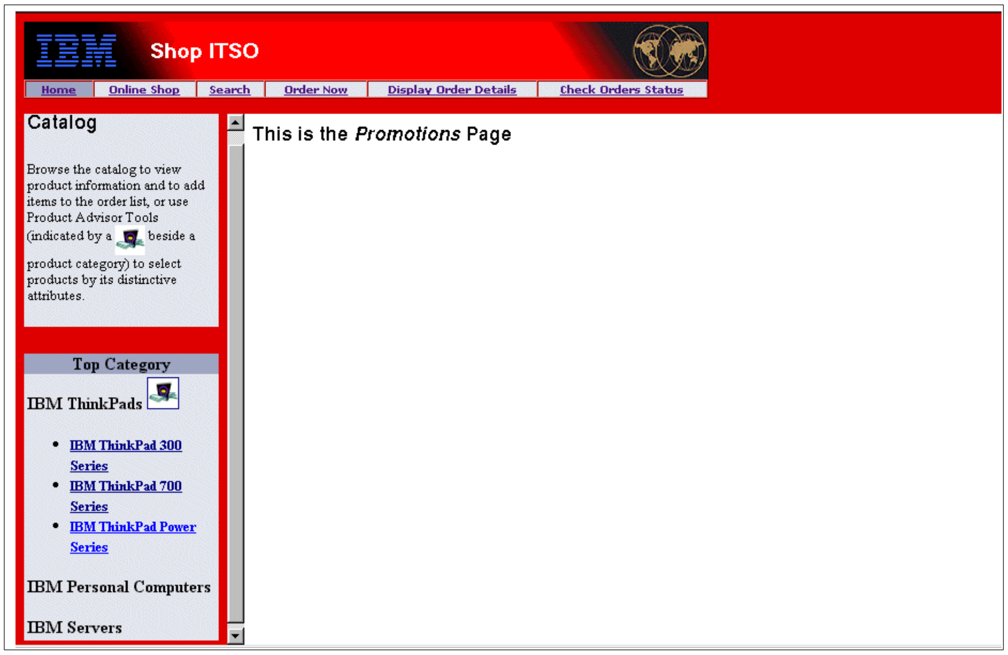
The Category Tree is shown in the left frame of a new window with navigation bar2, when the customer clicked the Online Shop from the navigation bar1. From this tree, the customer can navigate through our product catalog.

In the main frame of this page, we show our promotion page (an HTML page), which shows our special offerings of the week. From this promotion page, the customer can go directly to the product page of the chosen product.

The Category Tree (1) shows the second level categories, in our case, the IBM ThinkPads, IBM Personal Computers and the IBM Servers categories. The third level categories are shown directly under the second level categories. The first level category, the Top Category, is not shown in this page. For more information about this, see 13.15.1, “Net.Data Macro to Show the Category Tree” on page 243.

The Category Tree page is invoked from the navigation bar1 through the Net.Commerce Command *CategoryDisplay* (see Figure 24 on page 64). Through this command, Net.Commerce finds the Net.Data macro named cat0.d2w for the Home Category (Top Category, in our case).

Our first window in our store application ShopITSO is shown in Figure 25. You can see the three frames. The top frame shows the banner and navigation bar2. In the left frame, we show the catalog tree. In the main frame, we show our Promotion page (an HTML page), which shows our special offerings of the week.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

Top Category

IBM ThinkPads

- IBM ThinkPad 300 Series
- IBM ThinkPad 700 Series
- IBM ThinkPad Power Series

IBM Personal Computers

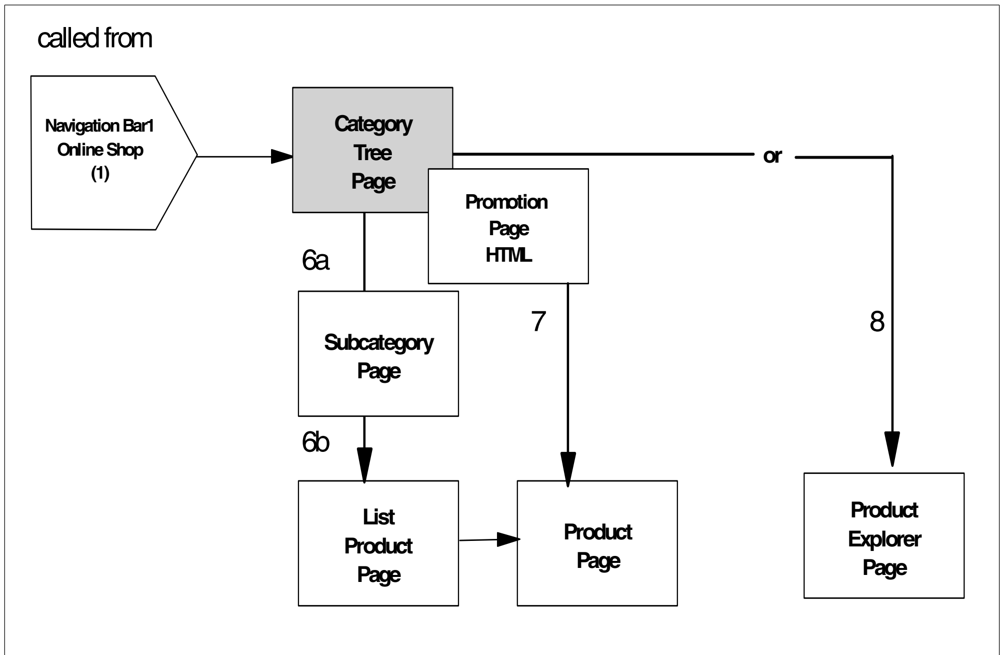
IBM Servers

This is the *Promotions* Page

Figure 25. Catalog Tree with Navigation Bar2 and Promotions Page

3.9.4.3 Navigation Flow from Online Shop

In Figure 26 on page 66, you find the navigation flow from our first window that is shown when the customer starts the Online Shop from navigation bar1 (our home page). Remember that the customer sees our catalog tree in the left frame first with the Promotions page in the main frame (see Figure 25).



6a = /cgi-bin/ncommerce3/CategoryDisplay?cgmenbr=$(MerchantRefNum)&cgrfnbr=$(V_crpcgnbr)
6b = /cgi-bin/ncommerce3/ProductDisplay?prrfnbr=$(V_PRRFNBR)&prmenbr=$(V_PRMENBR)
7 =/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=$(V_PRRFNBR)&prmenbr=$(V_PRMENBR)
8 = /servlet/icviewer/ca_html/shopitso_pe.html?cgrfnbr=$(V_CGRFNBR)&cgmenbr=$(MerchantRefNum)

Figure 26. Navigation Flow from Our Category Tree and Promotion Page

By clicking on the categories (category tree), the customer can navigate through our product catalog, from parent category to child category, until they reach the product list. These are the connections marked with 6a and 6b in Figure 26.

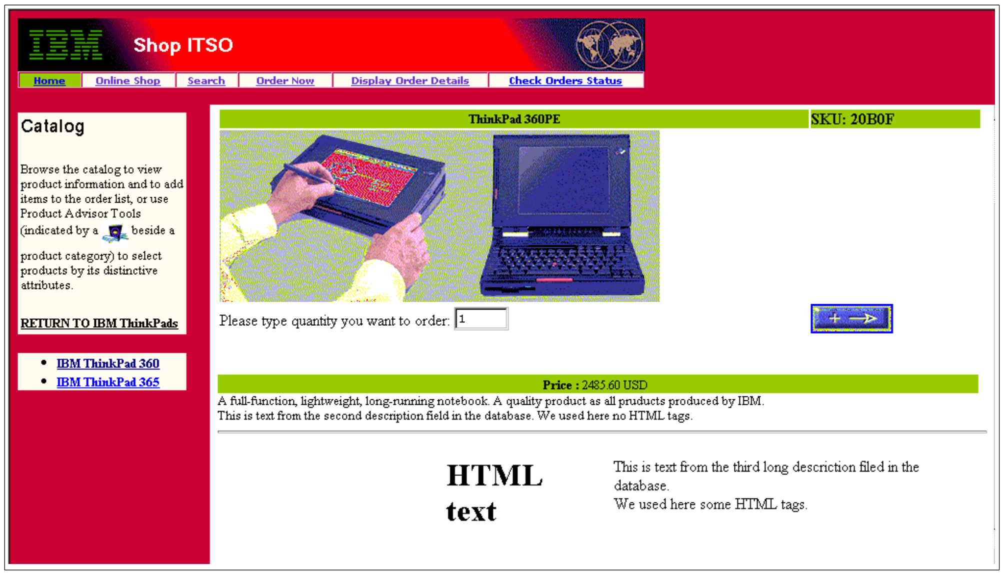
All of these pages are shown in the left frame of the window. In the main frame, the last page displayed is shown as unchanged. For example, in Figure 27 and Figure 28 on page 67, the last page shown in the main frame is a product page.

The Net.Commerce commands used to navigate are:

- `/cgi-bin/ncommerce3/`**`CategoryDisplay`**`?cgmenbr=merchant&cgrfnbr=$(V_crpcgnbr` as long as the child is a category and
- `/cgi-bin/ncommerce3/`**`ProductDisplay`**`?prrfnbr=$(V_PRRFNBR)&prmenbr=$(V_PRMENBR)` when the child is a product.

The Net.Data macros for these two display pages are cat1.d2w and prod1.d2w in our ShopITSO sample shop.

For an example of our subcategory window in the left frame, see Figure 26. As you can see here, it is possible to return to the subcategory from which the customer comes (the IBM ThinkPads, in this case).



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPads

- IBM ThinkPad 360
- IBM ThinkPad 365

ThinkPad 360PE | SKU: 20B0F

Please type quantity you want to order: 1

Price : 2485.60 USD

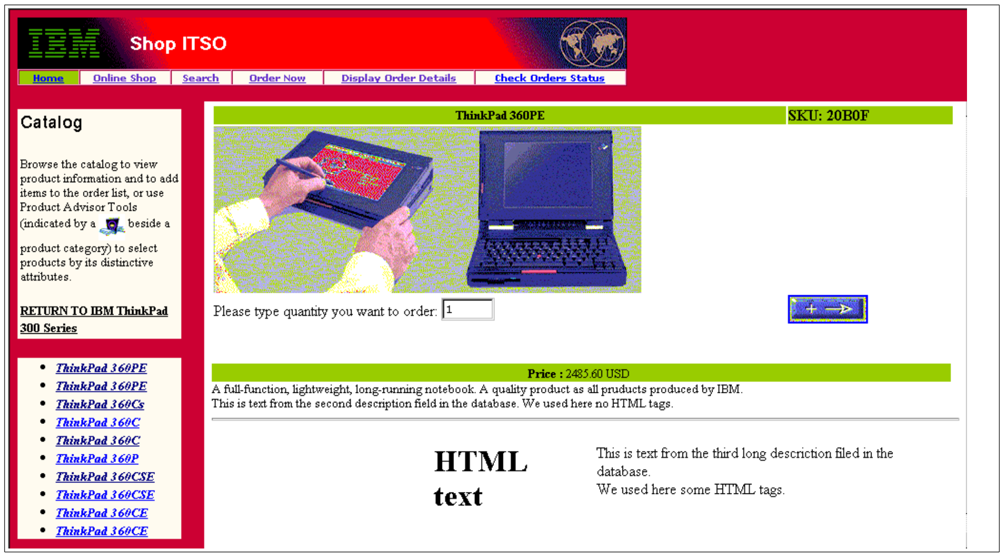
A full-function, lightweight, long-running notebook. A quality product as all pruducts produced by IBM.
This is text from the second description field in the database. We used here no HTML tags.

HTML text

This is text from the third long descriction filed in the database.
We used here some HTML tags.

Figure 27. Subcategory Page in Left Frame; Product Page in the Main Frame

You can see the product list page in the left frame of Figure 28. From here, the customer can return back to the category to which this product belongs.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

ThinkPad 360PE | SKU: 20B0F

Please type quantity you want to order: 1

Price : 2485.60 USD

A full-function, lightweight, long-running notebook. A quality product as all pruducts produced by IBM.
This is text from the second description field in the database. We used here no HTML tags.

HTML text

This is text from the third long descriction filed in the database.
We used here some HTML tags.

Figure 28. Product List Page in Left Frame; Product Page in the Main Frame

From the promotion page (7) in Figure 26, the customer can go directly to the product page of the selected product. The link used is the Net.Commerce command `ProductDisplay`.

To go to the Product Explorer page, the Net.Commerce command marked as (8) in Figure 26 is used in the link. For more information about Product Explorer, see Chapter 14, “Enhancing the Store Using Product Advisor” on page 275.

3.9.4.4 Navigation Flow from Product Explorer Page

From the Product Explorer page, a shopper goes to the product page and sees the description of the product.

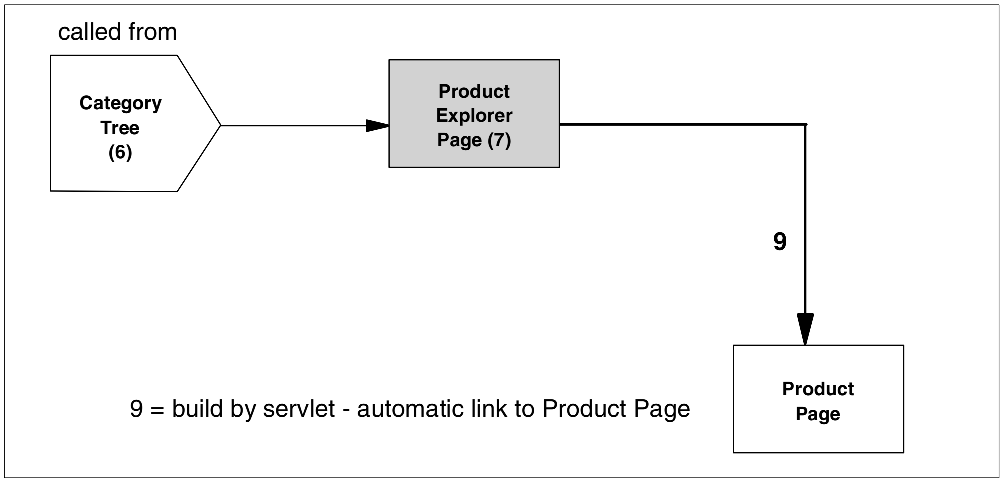

called from
Category Tree (6)
Product Explorer Page (7)
9
Product Page
9 = build by servlet - automatic link to Product Page

Figure 29. Navigation Flow from the Product Explorer Page

The link to the product page is generated by the servlet. For more information about this, see 14.1, “What a Product Advisor Is” on page 275.

3.9.4.5 Navigation Flow from the Product Page

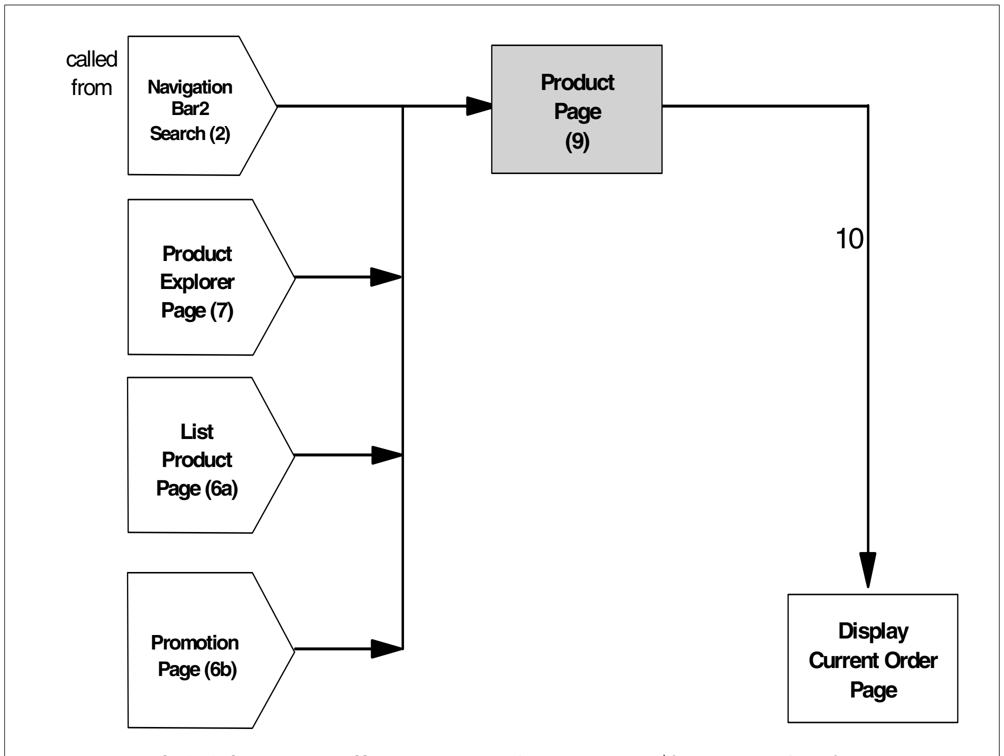
The following points describe our Product page:

- Content

  The Product page shows all of the information for this product such as the description of the product, a product image, the product price from the back-end system, and so on.

- Behavior

  From the Product page, a shopper can add a product to the current order. Next, they see the Display Current Order page. The information for the product, such as product reference number, quantity ordered and so on, is stored in the Net.Commerce database (SHIPTO).


called from
Navigation Bar2 Search (2)
Product Explorer Page (7)
List Product Page (6a)
Promotion Page (6b)
Product Page (9)
10
Display Current Order Page

10 = /cgi-bin/ncommerce3/OrderItemUpdate?merchant_rn =$(MerchantRefNum)
&product_rn =prrfnbr&quantity =Q&shipmode_rn=$(SHIPPING_MODE)
&url="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)"

*** (This command is coded as one long string. It is split up here across multiple lines so it would fit on the page) ***

Figure 30. Navigation Flow for Product Page

The content of the product page is implemented with the *product template*, in our case, the prod1.d2w Net.Data macro. This macro calls the Net.Commerce commands to implement the behavior.

Because we get the product price from our back-end system and not from the Net.Commerce product price tables, we have to customize this behavior in two places. The first place is in the product template. The other place is in the overridable function (OF) GetBaseUnitPrc. We have to replace the default overridable function GetBaseUnitPrc with a new OF that interfaces to the back-end system. The GetBaseUnitPrc OF is called by the GET_BASE_UNIT_PRC task, which is called by the `ProductDisplay` command. For more details, refer to 19.4, “Overridable Function by Example” on page 419.

For more information about product templates, see 13.7, “Creating the Product Catalog” on page 203. For more information about our Net.Data macro prod1.d2w, see 13.15.4, “Changes to Net.Data Macro for the Product Display” on page 253.

To add the product information in the database table SHIPTO, use the Net.Commerce command `OrderItemUpdate` with the parameters shown in Figure 30 on page 69 in combination with the `OrderItemDisplay` command. The value for the merchant_rn parameter comes from the include file. The value for product_rn comes from the PRODUCT table. The value for the quantity parameter comes from the HTML text form field *quantity*. The value for the shipmode_rn comes from the SHIPPING table. The value for the URL is the `OrderItemDisplay` command with the merchant_rn parameter.

The `OrderItemDisplay` command without the parameter addr_rn sets the SHIPTO_ASSOC view task to display a general shipping details page. If the parameter addr_rn is specified on the `OrderItemDisplay` command, the SHIPTO_DSP view task is set to display a specific shipping page.

The Net.Data macro, which is executed to show the next page in the e-business application (in this case, the Display Current Order page), is assigned for either the shop or the mall. Refer to 13.15.2, “Finding or Assigning a Net.Data Macro for a Specific Display” on page 244, for information about how to get the name of the macro assigned.

3.9.4.6 Navigation Flow from the Display Current Order Page

This section explains the Display Current Order page. The flow is shown in Figure 31 on page 71.

- Content

  For a sample of the Display Current Order page, see Figure 19 on page 60.

  The content of the Display Current Order page is implemented with the Net.Data macro shipto.d2w. More information about our Net.Data macro shipto.d2w is contained in 13.15, “Modifying Net.Data Macros” on page 241. This macro calls Net.Commerce commands to implement the behavior.

  Because we get the product price from our back-end system and not from the Net.Commerce product price tables, we have to customize this behavior. We have to replace the default Overridable Function (OF) GetBaseSpePrc with a new OF that interfaces to the back-end system. The GetBaseSpePrc OF is called by the GET_BASE_SPE_PRC task, which is called by the `OrderItemDisplay` command (see 19.4, “Overridable Function by Example” on page 419).

- Behavior

  From the Display Current Order page, a shopper can change the ordered quantity for a product, delete a product, or submit the order.

  When they change the quantity or delete a product, the next window that appears is the Display Current Order page recalculated (14). When they submit the order, the next window that appears is the Order Accepted page (11).

  The Address Form (12) and the Payment Form (13) are not separate pages. They are included HTML forms in the Order Accepted page (11) . The behavior is implemented in the Order Accepted page (see 3.9.4.7, “Navigation Flow from Order Accepted Page” on page 73).

  The information for a changed product (quantity or delete) has to be updated in the Net.Commerce SHIPTO database table.

  When the customer submits the order, the Net.Commerce ORDERS and ORDERPAY tables are updated.

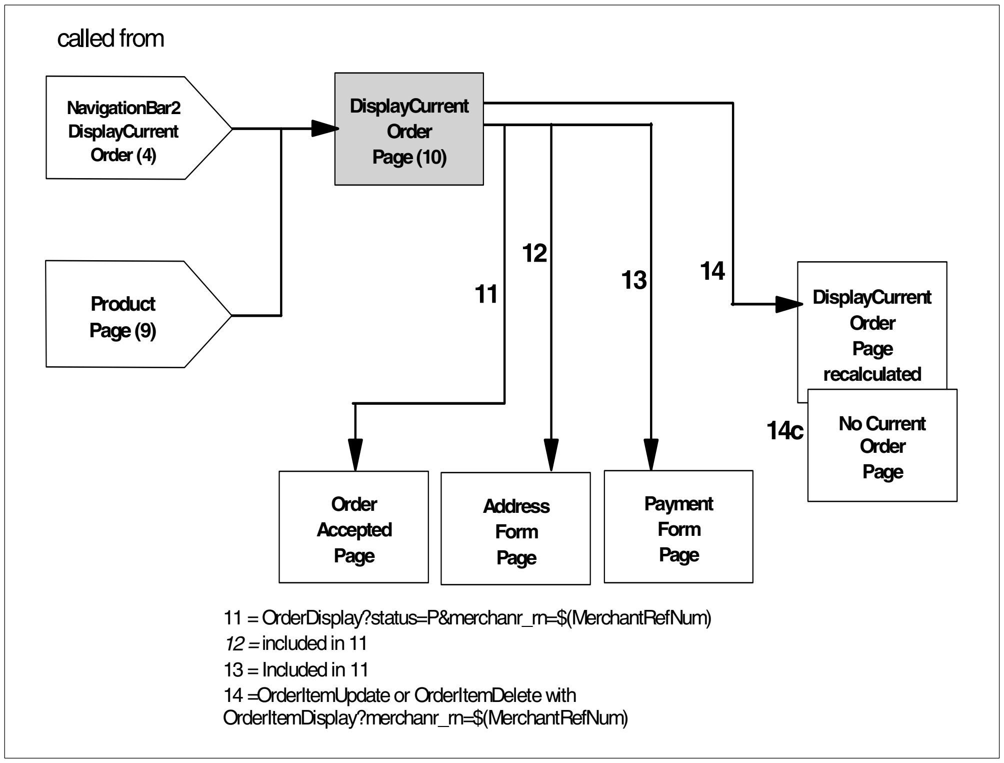

called from
NavigationBar2 DisplayCurrent Order (4)
Product Page (9)
DisplayCurrent Order Page (10)
11
12
13
14
DisplayCurrent Order Page recalculated
14c
No Current Order Page
Order Accepted Page
Address Form Page
Payment Form Page
11 = OrderDisplay?status=P&merchanr_rn=$(MerchantRefNum)
12 = included in 11
13 = Included in 11
14 =OrderItemUpdate or OrderItemDelete with OrderItemDisplay?merchanr_rn=$(MerchantRefNum)

Figure 31. Navigation Flow from Our Display Current Order Page

To change product information in the SHIPTO database table when the quantity has been changed by the customer, use the Net.Commerce command `OrderItemUpdate` with the parameters shown in the following example in combination with the `OrderItemDisplay` command:

```
/cgi-bin/ncommerce3/OrderItemUpdate?merchant_rn =$(MerchantRefNum)
&shipto_rn =$(V_strfnbr)&quantity=quantity&shipmode_rn=$(V_stsmnbr)
&url="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)
```

The value for the merchant_rn parameter comes from the include file. The value for shipto_rn comes from the SHIPTO table. The value for quantity comes from the HTLM text form field quantity. The value for the URL is the `OrderItemDisplay` command with the merchant_rn parameter.

To delete a row in the SHIPTO database table when the product is deleted from the order by the customer, use the Net.Commerce command `OrderItemDelete` with the parameter shown in combination with the `OrderItemDisplay` command:

```
/cgi-bin/ncommerce3/OrderItemDelete?&shipto_rn =$(V_strfnbr)
&url="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)
```

The value for the merchant_rn parameter comes from the include file. The value for the shipto_rn comes from the SHIPTO table. The value for the URL is the `OrderItemDisplay` command with parameter merchant_rn.

If there are no more items in the SHIPTO table for the customer (for example, all items were deleted), you should tell the customer. This situation is marked with 14c in our navigation flow.

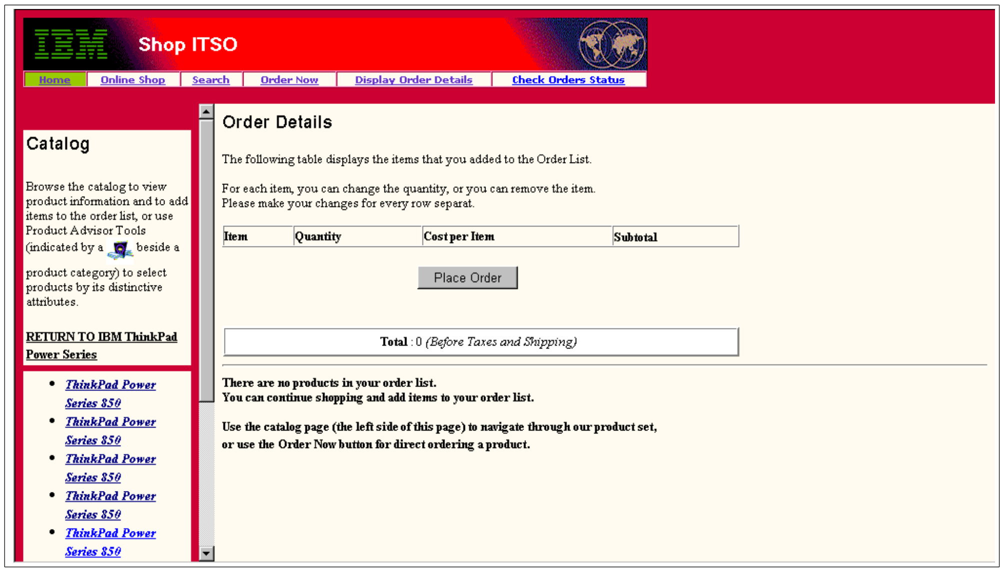
For both update and delete, the `OrderItemDisplay` command is responsible for showing the next page in the e-business application (in this case, the Display Current Order page), as described in 3.9.4.5, “Navigation Flow from the Product Page” on page 68. The `OrderItemDisplay` command has no error condition defined. Your Net.Data macro, which creates this page, should take care of error conditions.

You can use the SQL functions message block where the results of non-zero SQL return codes are processed to handle error conditions. The most common use for this message block is to process the SQL code 100 where no rows were returned from the query. You can also use a default message section that processes all possible error codes in the same manner as we did in our shipto.d2w macro.

The following code fragment shows the error condition processing in our shipto.d2w macro:

```
%function(dtw_odbc) GET_TOTAL_DETAILS() {
SELECT stprice, stquant, stcpcur, strfnbr, stsanbr, ststat, stsmnbr,
prsdesc, prrfnbr, prnbr
FROM shipto, product
WHERE stmenbr=$(MerchantRefNum) and stshnbr=$(SHOPPER_REF) and stprnbr=prrfnbr
and ststat='P'
   %REPORT{
%ROW{
%}
%}
%MESSAGE{
     default: {
<FONT SIZE=2><HR><b>There are no products in your order list. <br>
You can continue shopping and add items to your order list. <br><br>
   Use the catalog page (the left side of this page) to navigate
through our product set,<br>
   or use the
Order Now button for direct ordering a product.</B></FONT>
%}:continue
%}
%}
```

The page that is built by this error code (14c) is shown in Figure 32 on page 73.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad Power Series

- ThinkPad Power Series 850
- ThinkPad Power Series 850
- ThinkPad Power Series 850
- ThinkPad Power Series 850
- ThinkPad Power Series 850

Order Details

The following table displays the items that you added to the Order List.

For each item, you can change the quantity, or you can remove the item.
Please make your changes for every row separat.

Item	Quantity	Cost per Item	Subtotal

Place Order

Total : 0 *(Before Taxes and Shipping)*

There are no products in your order list.
You can continue shopping and add items to your order list.

Use the catalog page (the left side of this page) to navigate through our product set, or use the Order Now button for direct ordering a product.

Figure 32. No Current Order Page

Use the Net.Commerce command `OrderDisplay` with the following parameter to insert a new order in the ORDER table (status = P) and the ORDERPAY table when the customer submits the order (11):

```
/cgi-bin/ncommerce3/OrderDisplay?status=P&merchant_rn =$(MerchantRefNum)
```

The value for the merchant_rn parameter comes from the include file and the value for status is “P”, the status for pending order.

The `OrderDisplay` Net.Commerce command has a huge number of tasks that are called. Refer the handbook *Net.Commerce for AS/400 Commands, Tasks, Overridable Functions, and Database Tables* (dbtodcmd.pdf), for more information about this and other Net.Commerce commands.

One of the tasks called is the ORD_DSP_PEN *view task* to display the pending order (our next page Order Accepted page (11)). We should not use a URL for showing the next page in the e-business application. The name of the Net.Data macro that is executed by the ORD_DSP_PEN task is order.d2w. Refer to 13.15.2, “Finding or Assigning a Net.Data Macro for a Specific Display” on page 244, for information about how to obtain the name of the macro assigned.

If no orders match the specified parameters, the `OrderDisplay` command sets the `ORD_NONE` error task to handle the error. In our case, we assigned the ordnone.d2w macro to this task to keep the customer informed. Refer to 13.16.2, “Assigning a Net.Data Macro to an Exception Task” on page 261, for information about how to assign a macro to an error task.

3.9.4.7 Navigation Flow from Order Accepted Page

This section describes our Order Accept process. The flow is shown in Figure 31 on page 71.

- Content

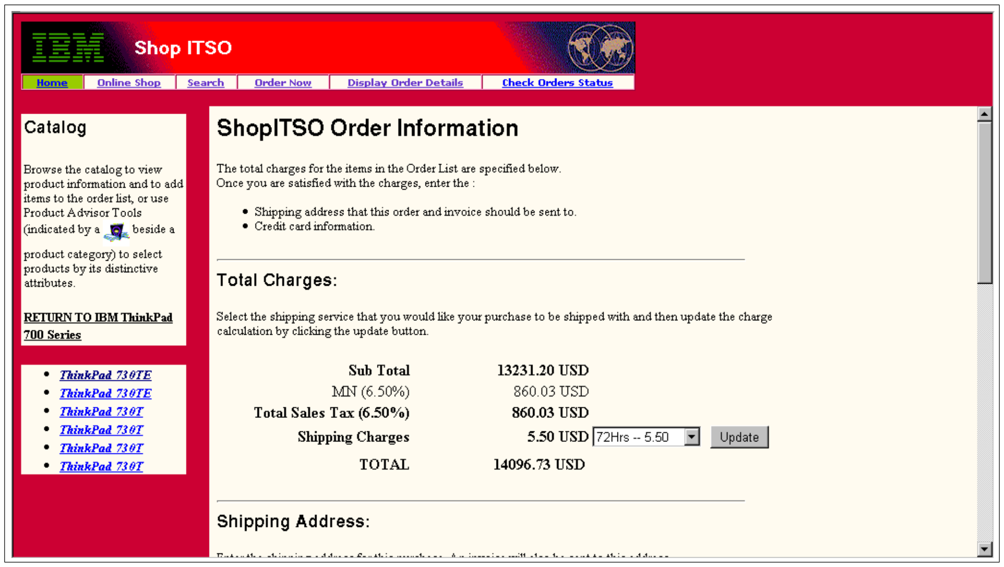
  The Order Accepted page is divide in three areas. It is one page that is scrollable.

The first area shows the total values of the entire order such as order total amount before taxes, tax, shipping charges, and the total order amount. In this area, the customer can choose the shipping method in a selection list. When they use the Update button, the selected shipping method with the corresponding shipping charges are updated.

The second area shows the Address Form (12) in which the customer types their address information including the e-mail address.

In the third area, the customer can choose if they want to pay with a credit card with or without SET protocol (13). If they choose payment without SET protocol, they have to type in credit card information such as credit card type, card number, expiration month, and expiration year. If they choose to pay with SET, Net.Commerce sends a wake-up message to the eWallet of the customer's browser.

Figure 33 shows the first area of the Order Accepted page.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- ThinkPad 730TE
- ThinkPad 730TE
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T

ShopITSO Order Information

The total charges for the items in the Order List are specified below.
Once you are satisfied with the charges, enter the :

- Shipping address that this order and invoice should be sent to.
- Credit card information.

Total Charges:

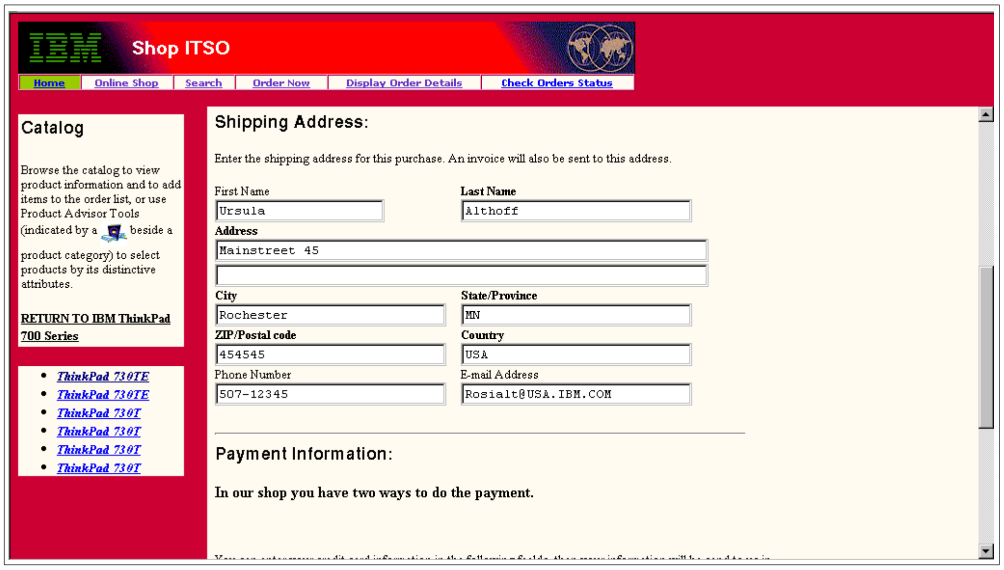
Select the shipping service that you would like your purchase to be shipped with and then update the charge calculation by clicking the update button.

Sub Total	13231.20 USD
MN (6.50%)	860.03 USD
Total Sales Tax (6.50%)	860.03 USD
Shipping Charges	5.50 USD 72Hrs -- 5.50
TOTAL	14096.73 USD

Shipping Address:

Figure 33. Order Accepted Page — First Area

Figure 34 on page 75 shows the second area of the Order Accepted page.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- ThinkPad 730TE
- ThinkPad 730TE
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T

Shipping Address:

Enter the shipping address for this purchase. An invoice will also be sent to this address.

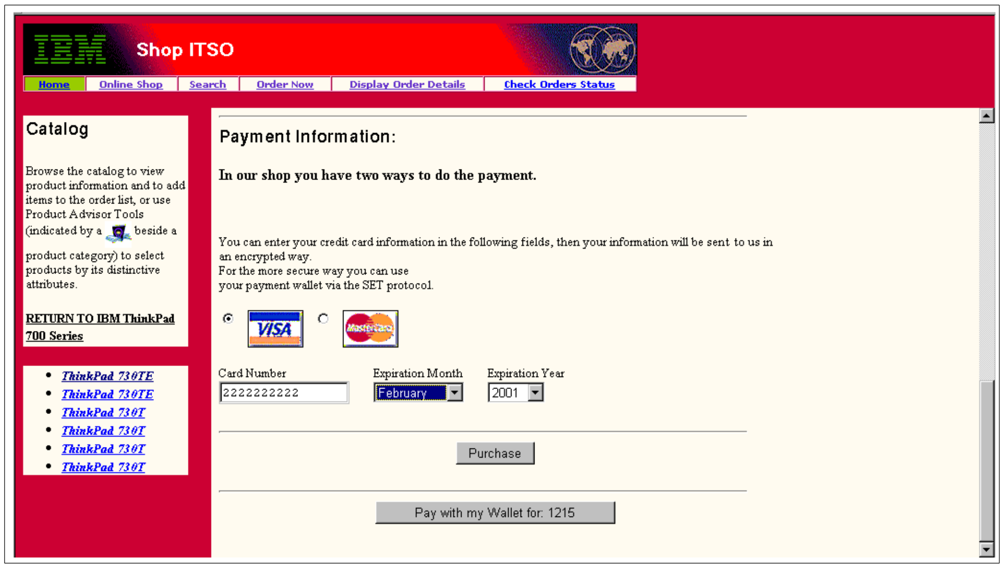
First Name: Ursula
Last Name: Althoff
Address: Mainstreet 45
City: Rochester
State/Province: MN
ZIP/Postal code: 454545
Country: USA
Phone Number: 507-12345
E-mail Address: Rosialt@USA.IBM.COM

Payment Information:

In our shop you have two ways to do the payment.

Figure 34. Order Accepted Page — Second Area

Figure 35 shows the third area of the Order Accepted page.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- ThinkPad 730TE
- ThinkPad 730TE
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T

Payment Information:

In our shop you have two ways to do the payment.

You can enter your credit card information in the following fields, then your information will be sent to us in an encrypted way.
For the more secure way you can use your payment wallet via the SET protocol.

[✓ VISA] MasterCard

Card Number: 2222222222
Expiration Month: February
Expiration Year: 2001

Figure 35. Order Accepted Page — Third Area

The content of the Order Accepted page is implemented with the Net.Data macro order.d2w. This macro calls the Net.Commerce commands to implement the behavior. Refer to Figure 36 on page 76 during this discussion.

- Behavior

  From the Order Accepted page, a shopper places their order including their address, which shipmode they want for the whole order, and how they want to make the payment. From this page, the shopper places the final order.

  If the order can be accepted (address is filled in, payment information is verified, or in case of payment with SET the request authorization is done successful), the customer gets the Order Confirmation page (15).

For more information about payment processing, see 16.2, “Payment Server Payment Processing” on page 366.

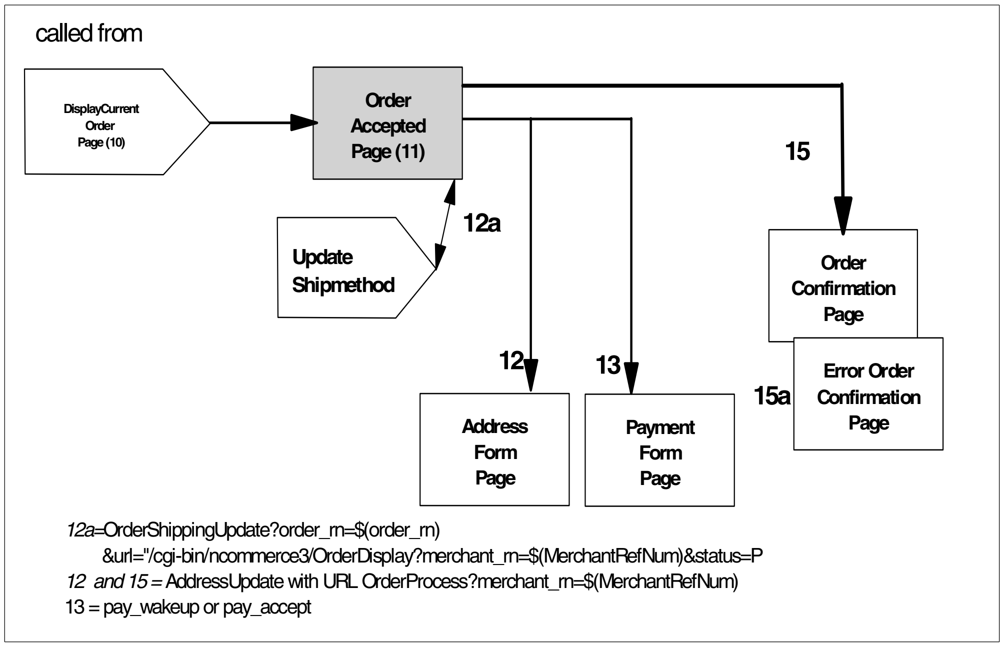

called from
DisplayCurrent Order Page (10)
Order Accepted Page (11)
Update Shipmethod
12a
15
Order Confirmation Page
12
13
15a
Error Order Confirmation Page
Address Form Page
Payment Form Page
12a=OrderShippingUpdate?order_rn=$(order_rn)
&url="/cgi-bin/ncommerce3/OrderDisplay?merchant_rn=$(MerchantRefNum)&status=P
12 and 15 = AddressUpdate with URL OrderProcess?merchant_rn=$(MerchantRefNum)
13 = pay_wakeup or pay_accept

Figure 36. Navigation Flow from Our Order Accepted Page

To change the shipmode (the shipping provider with the assigned shipping cost) for the whole order (12a), use the Net.Commerce command `OrderShippingUpdate` with the following parameters. The customer sees the same page again with the new calculated amount for the shipping cost.

```
/cgi-bin/ncommerce3/OrderShippingUpdate?order_rn =$(order_rn)
&url="/cgi-bin/ncommerce3/OrderDisplay?merchant_rn=$(MerchantRefNum)
&status=P
```

The second area of the Order Accepted page shows the input fields for the address. The third area shows the input and selection fields for the credit card information. When the customer wants to pay *without using a wallet* (without using SET), they fill in the required values in these credit card fields.

To store the address for the whole order (12) and to verify the payment information values (13), use the Net.Commerce command `AddressUpdate` with the following parameters:

```
/cgi-bin/ncommerce3/AddressUpdate?&sarfnbr=$(V_sarfnbr)&ALL ADDRESS
FIELDS&sanick=$(SESSION_ID)&=order_rn=$(order_rn)&merchant_rn=$(MerchantRefNum
)&cctype=$(cctype)&ccxmonth=$(ccxmonth)&cxyecar=$(cxyyecar)&url=/cgi-bin/ncomm
erce3/OrderProcess?merchant_rn=$(MerchantRefNum)
```

The values for *all address fields* come from the HTML form input fields, which are typed in by the customer.

You can also use your payment gateway to request the capture of a payment from the acquirer after the order is fulfilled (shipped), even if the customer does not use SET. To do this, the customer fills in the information for the credit card that is being used for payment (as before). You use the Net.Commerce command pay_accept, instead of the OrderProcess command, in addition to the AddressUpdate command with the same parameters as described before.

```
/cgi-bin/ncommerce3/AddressUpdate?&sarfnbr=$(V_sarfnbr)&ALL ADDRESS
FIELDS&sanick=$(SESSION_ID)&=order_rn=$(order_rn)&merchant_rn=$(MerchantRefNum
)&cctype=$(cctype)&ccxmonth=$(ccxmonth)&cxyecar=$(cxyyecar)&url=pay_accept?m
erchant_rn=$(MerchantRefNum)&order_rn=$(order_rn)
```

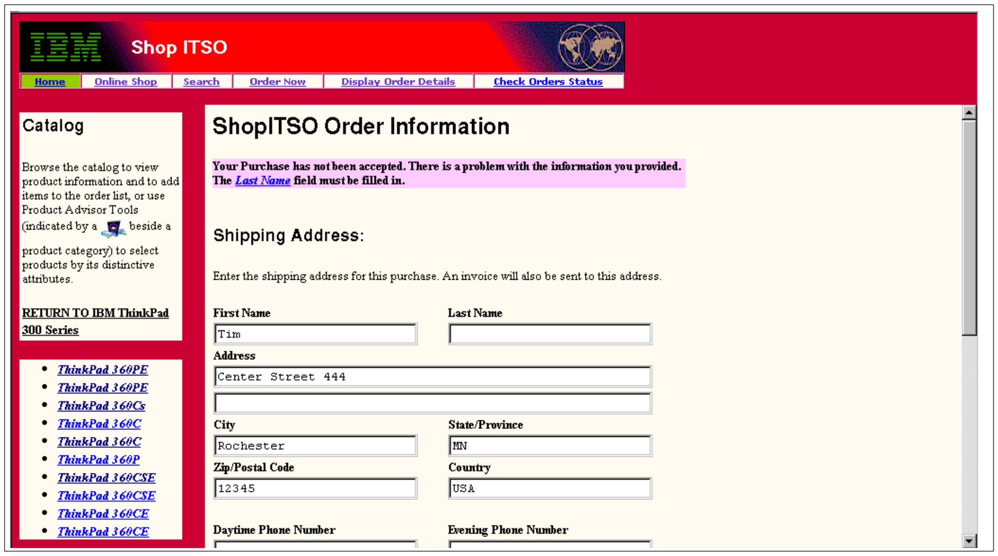
This is the *Merchant Originated Payment* process. See 6.2.1, “Merchant Originated Payment” on page 96, for the details.

When the customer wants to pay *with their wallet*, the Net.Commerce command /pay_wakeup?order_rn=$(order_rn)&merchant_rn=$(MerchantRefNum) is used. In this case, the credit card information fields (third area) do not have to be filled in. This is because all the information is based on the SET certificate of the customer.

For more information about Payment, refer to 16.2, “Payment Server Payment Processing” on page 366.

In case of error conditions (15a), for example the name in the customer address was not filled in, the AddressUpdate command sets the BAD_ADRBK_MODIFY error task. We assigned the err_adrbk_up.d2w macro to this error task to show the Error Order Confirmation page to the customer. The first part of this page is shown in Figure 37. In the err_adrbk_up.d2w macro, we used the same Net.Commerce commands as for the Order Accepted page, with the exception of the OrderShippingUpdate command. This is because the shipmode cannot be changed again from this page.

After the order process is successfully completed (address is typed in and verified and payment information is valid), the customer sees the next window with the acknowledgement of the order (Order Confirmation 15). This is done by the ORD_OK view task from the OrderProcess command. The Net.Data macro for this page is orderok.d2w.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

ShopITSO Order Information

Your Purchase has not been accepted. There is a problem with the information you provided.
The Last Name field must be filled in.

Shipping Address:

Enter the shipping address for this purchase. An invoice will also be sent to this address.

First Name	Last Name
Tim	

Address
Center Street 444

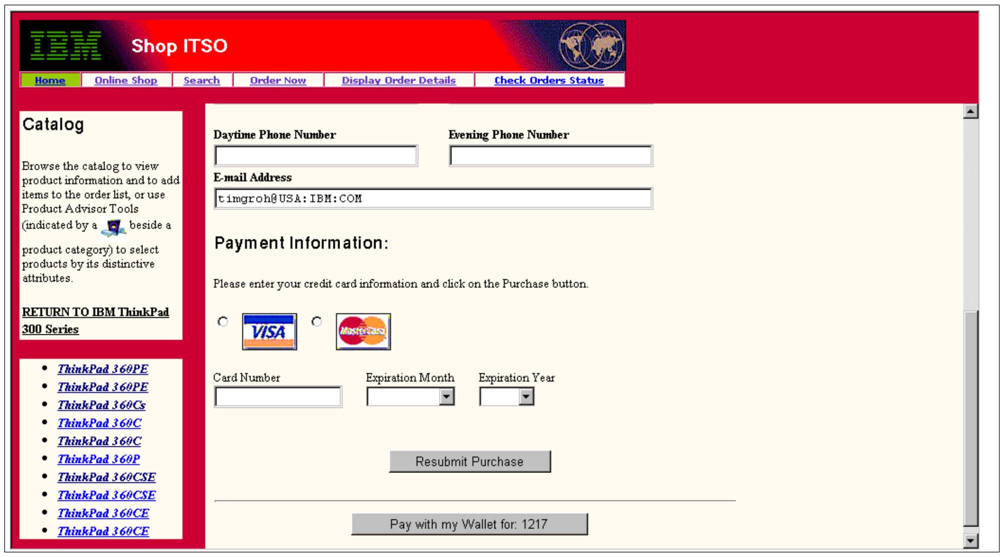
City	State/Province
Rochester	MN

Zip/Postal Code	Country
12345	USA

Daytime Phone Number
Evening Phone Number

Figure 37. Error Order Confirmation Page (Part 1 of 2)

Figure 38 on page 78 shows part 2 of the Error Order Confirmation page.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

Daytime Phone Number

Evening Phone Number

E-mail Address: timgroh@USA:IBM:COM

Payment Information:

Please enter your credit card information and click on the Purchase button.

Card Number

Expiration Month

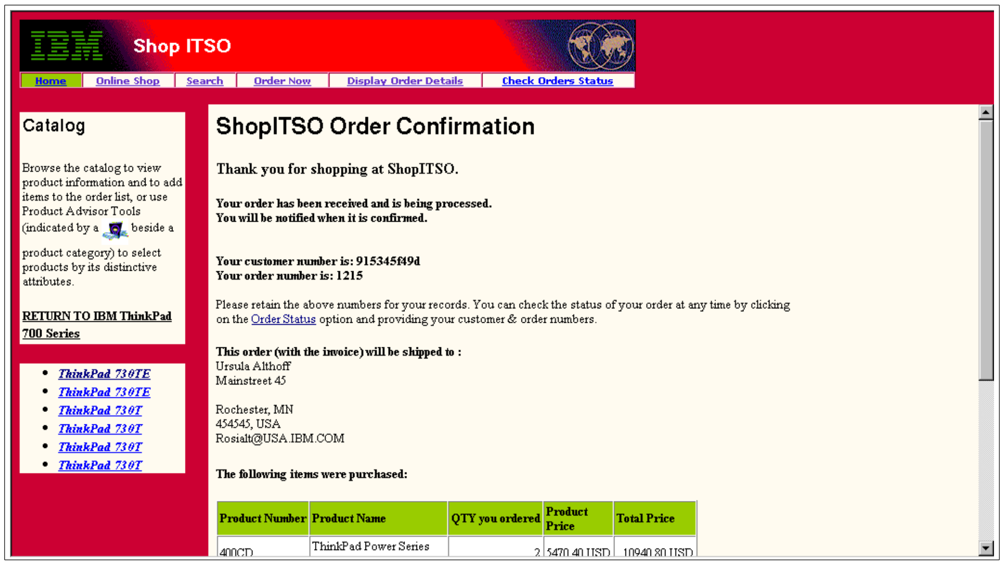
Expiration Year

Figure 38. Error Order Confirmation Page (Part 2 of 2)

3.9.4.8 Order Confirmation Page

Figure 39 through Figure 41 show our Order Confirmation page. It is one page that is scrollable.

In part 1 of the page (Figure 39), we tell the customer the assigned customer number (because we do not have customer registration and do not work with permanent customer numbers), the assigned order number, and the shipping address.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- ThinkPad 730TE
- ThinkPad 730TE
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T
- ThinkPad 730T

ShopITSO Order Confirmation

Thank you for shopping at ShopITSO.

Your order has been received and is being processed.
You will be notified when it is confirmed.

Your customer number is: 915345f49d
Your order number is: 1215

Please retain the above numbers for your records. You can check the status of your order at any time by clicking on the Order Status option and providing your customer & order numbers.

This order (with the invoice) will be shipped to :
Ursula Althoff
Mainstreet 45

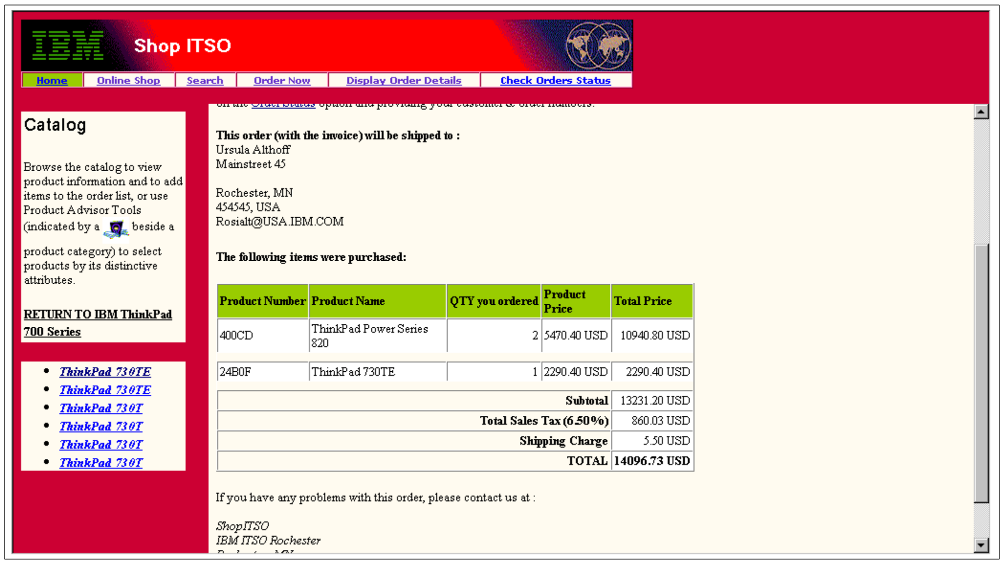
Rochester, MN
454545, USA
Rosialt@USA.IBM.COM

The following items were purchased:

Product Number	Product Name	QTY you ordered	Product Price	Total Price
400CD	ThinkPad Power Series	2	5470.40 USD	10940.80 USD

Figure 39. Order Confirmation Page (Part 1 of 3)

Figure 40 shows part 2 of the Order Confirmation page. Here, we show the products ordered, the quantity ordered, the product price, the extended price, and the total amount of the order.



on the Order Status option and providing your customer & order numbers.

Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- *ThinkPad 730TE*
- *ThinkPad 730TE*
- *ThinkPad 730T*
- *ThinkPad 730T*
- *ThinkPad 730T*
- *ThinkPad 730T*

This order (with the invoice) will be shipped to :
Ursula Althoff
Mainstreet 45

Rochester, MN
454545, USA
Rosialt@USA.IBM.COM

The following items were purchased:

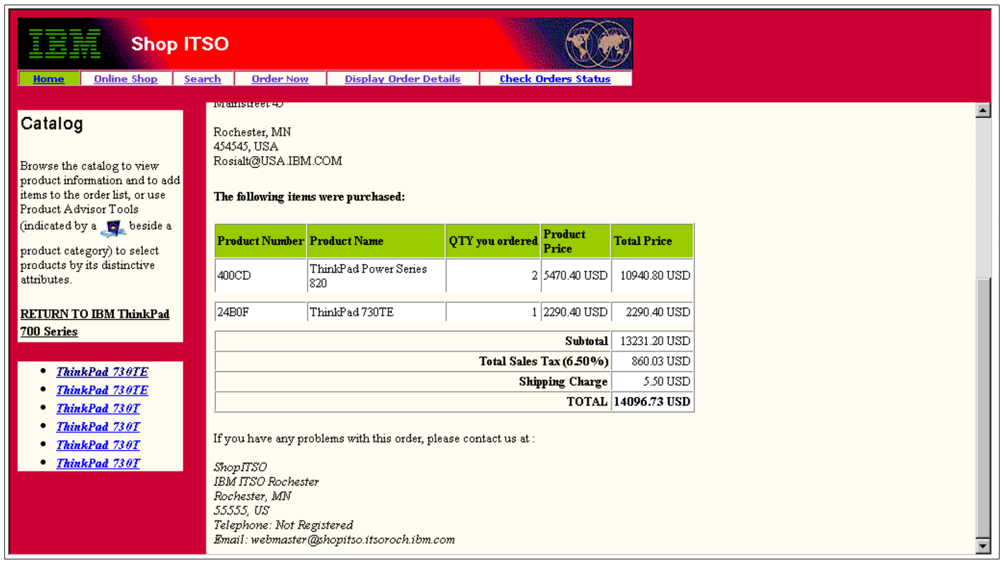
Product Number	Product Name	QTY you ordered	Product Price	Total Price
400CD	ThinkPad Power Series 820	2	5470.40 USD	10940.80 USD
24B0F	ThinkPad 730TE	1	2290.40 USD	2290.40 USD
			Subtotal	13231.20 USD
			Total Sales Tax (6.50%)	860.03 USD
			Shipping Charge	5.50 USD
			TOTAL	**14096.73 USD**

If you have any problems with this order, please contact us at :

ShopITSO
IBM ITSO Rochester

Figure 40. Order Confirmation Page (Part 2 of 3)

Part 3 of the Order Confirmation page shows our contact address, which comes from the Net.Commerce database.



Mainstreet 45

Rochester, MN
454545, USA
Rosialt@USA.IBM.COM

Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- *ThinkPad 730TE*
- *ThinkPad 730TE*
- *ThinkPad 730T*
- *ThinkPad 730T*
- *ThinkPad 730T*
- *ThinkPad 730T*

The following items were purchased:

Product Number	Product Name	QTY you ordered	Product Price	Total Price
400CD	ThinkPad Power Series 820	2	5470.40 USD	10940.80 USD
24B0F	ThinkPad 730TE	1	2290.40 USD	2290.40 USD
			Subtotal	13231.20 USD
			Total Sales Tax (6.50%)	860.03 USD
			Shipping Charge	5.50 USD
			TOTAL	**14096.73 USD**

If you have any problems with this order, please contact us at :

ShopITSO
IBM ITSO Rochester
Rochester, MN
55555, US
Telephone: Not Registered
Email: webmaster@shopitso.itsoroch.ibm.com

Figure 41. Order Confirmation Page (Part 3 of 3)

3.9.4.9 Navigation Flow Check Order Status Page

The Check Order Status page is called from the navigation bar2 and shows the status of a single order (see Figure 42 on page 80). The customer has to type in the customer and order number to get this information. Otherwise, they get an error page.

Because our back-end system updates the status of an order (for example shipping in process), we can display this information here. We use one Net.Data macro, which selects the data from the Net.Commerce database. This macro is called by the Net.Commerce command

`ExecMacro/$(STORENAME)/orderlstc.d2w/input`, from the navigation bar. Inside this macro, the `oderlstc.d2w/report` command is called to show the result.

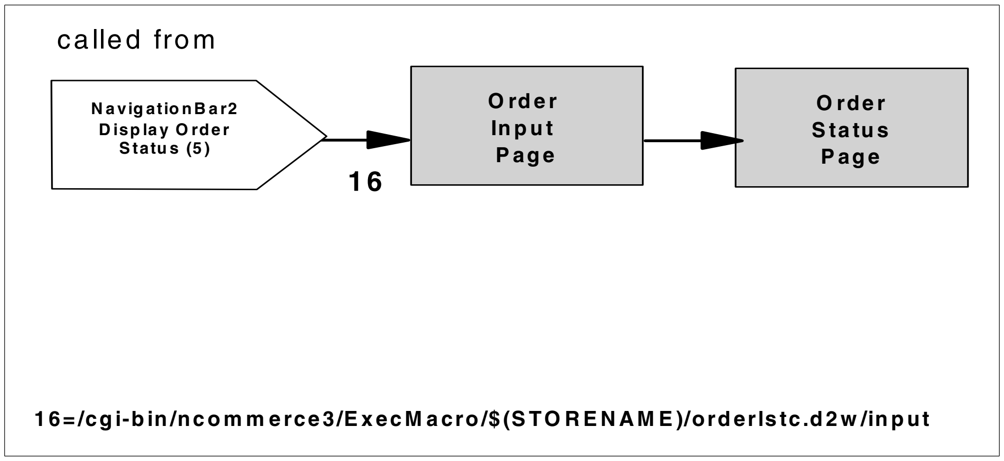

called from
NavigationBar2 Display Order Status (5)
16
Order Input Page
Order Status Page
16=/cgi-bin/ncommerce3/ExecMacro/$(STORENAME)/orderlstc.d2w/input

Figure 42. Navigation Flow to Check Order Status Page

3.9.4.10 Navigation Flow Search Results Page

In our search function, a customer can search for a product by its description. This is only a simple search function to illustration how to use Net.Data to build in functions. In our case, we used the @DTW_rTRANSLATE to translate the input search argument to uppercase and an SQL translate to translate the left side of a LIKE (the value in the database) to uppercase. Because the values in our product table are in mixed case, we must translate both values to a common case or the LIKE function will not work.

Our code extract of our searchrslt.d2w is shown in the following example:

```
%function(dtw_odbc) SEARCH_PRODUCTS() {
   select PRODUCT.PRRFNBR, PRODUCT.PRNBR, PRSDESC   from PRODUCT
   where prpub=1 and prmenbr=$(MerchantRefNum) and (translate(prsdesc) like
   '%@DTW_rTRANSLATE($(search))%')
%REPORT{
```

The demomall has more complex search functions integrated into it. Look at these samples to get other ideas of how to create search pages.

In our ShopITSO sample, the customer enters a search word in the first window (the HTML page). In the next window, they get the result. For the navigation flow, see Figure 43 on page 81.

The result can be more than one product. From the Search Results page, a shopper can choose a product and get to the corresponding Product page.

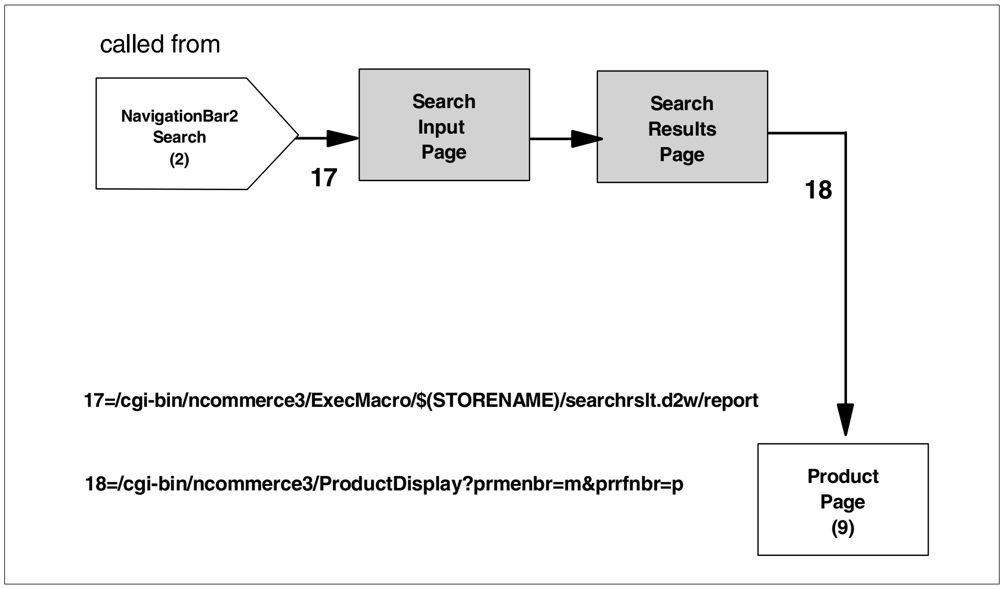
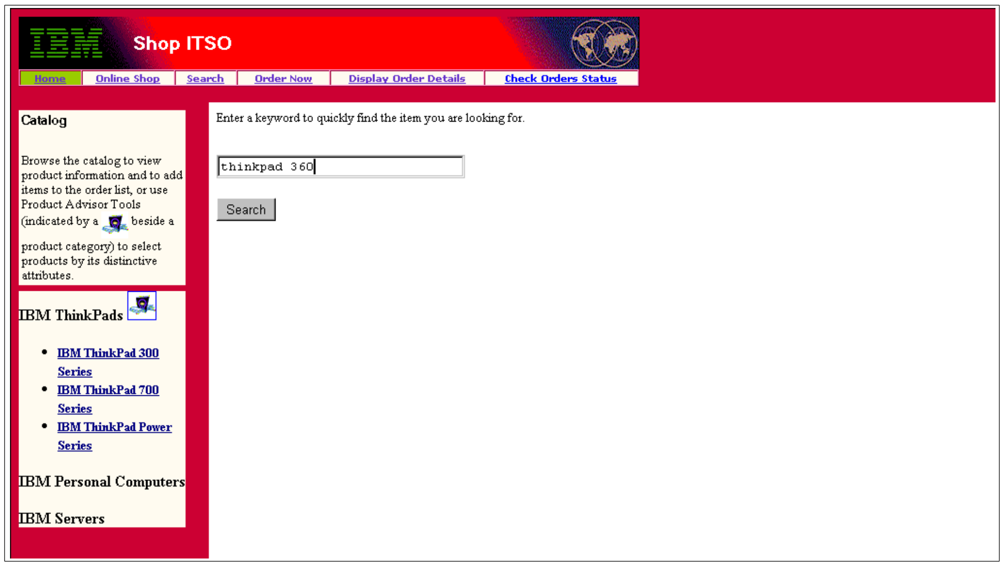

called from
NavigationBar2 Search (2)
17
Search Input Page
Search Results Page
18
17=/cgi-bin/ncommerce3/ExecMacro/$(STORENAME)/searchrslt.d2w/report
18=/cgi-bin/ncommerce3/ProductDisplay?prmenbr=m&prrfnbr=p
Product Page (9)

Figure 43. Navigation Flow for Search Page

The Search Input page is an HTML page `search.html` (17), called from the navigation bar2. This HTML page uses the Net.Commerce command `ExecMacro` to call the Net.Data macro `searchrst.d2w/report`, which shows the Search Result page.

Figure 44 shows the search HTML page.

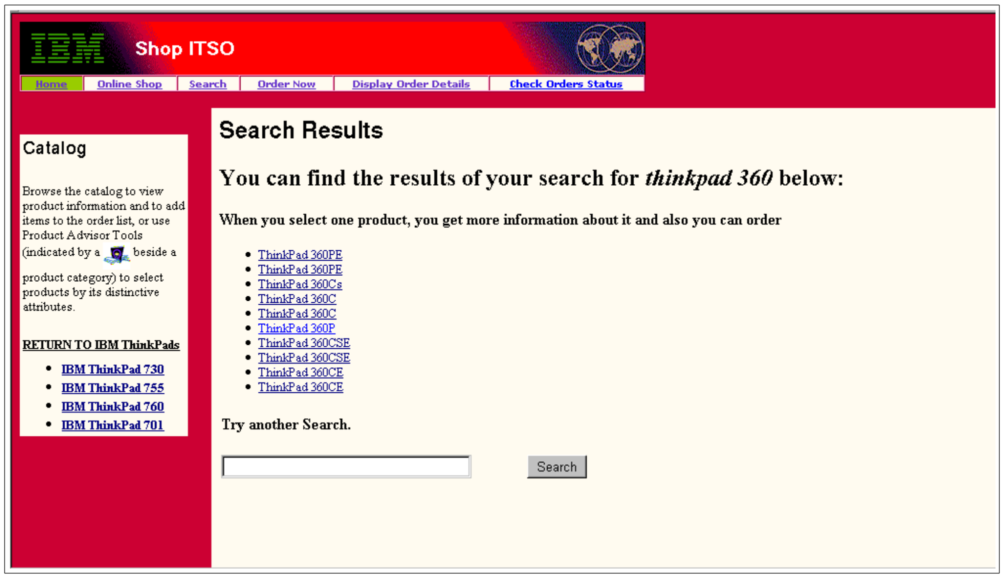


Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

Enter a keyword to quickly find the item you are looking for.

Figure 44. Search Input HTML Page

In the Net.Data macro, we use the Net.Commerce command `ProductDisplay` to get the Product page for the chosen product. Figure 45 on page 82 shows our Search Result page in our ShopITSO sample store.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPads

- IBM ThinkPad 730
- IBM ThinkPad 755
- IBM ThinkPad 760
- IBM ThinkPad 701

Search Results

You can find the results of your search for ***thinkpad 360*** below:

When you select one product, you get more information about it and also you can order

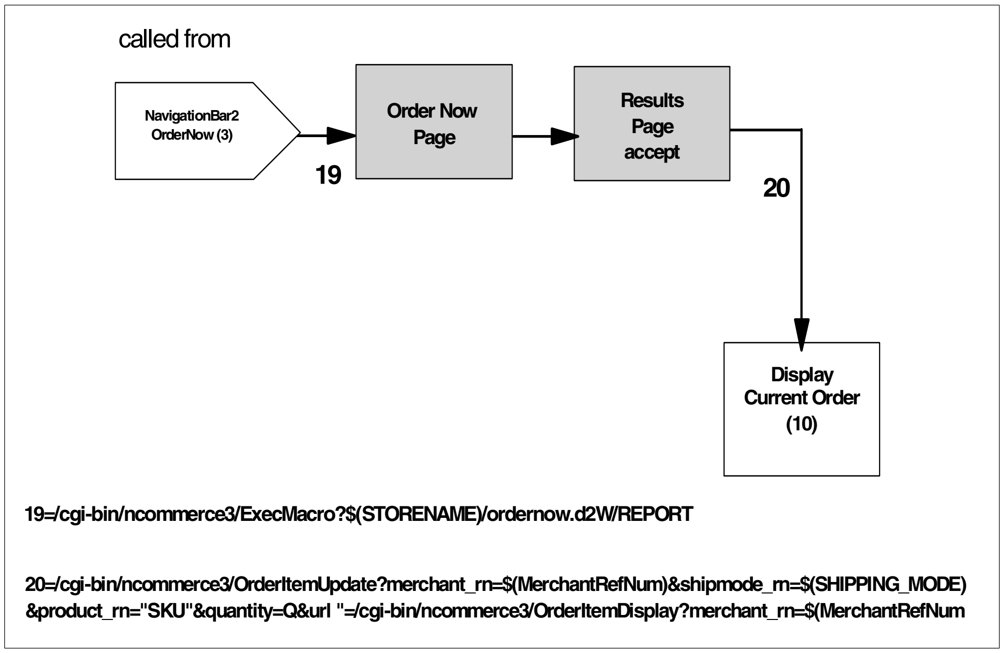
- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

Try another Search.

Figure 45. ShopITSO Search Result Page

3.9.4.11 Navigation Flow Order Now Page

In the Order Now page, a customer can enter a product SKU number together with a quantity they want to order. If the product is found, a result page is shown to the customer with the product description and the values entered for the customer. When the customer can accept the order, the Display Current Order page appears as shown in Figure 46.



```
19=/cgi-bin/ncommerce3/ExecMacro?$(STORENAME)/ordernow.d2W/REPORT

20=/cgi-bin/ncommerce3/OrderItemUpdate?merchant_rn=$(MerchantRefNum)&shipmode_rn=$(SHIPPING_MODE)
&product_rn="SKU"&quantity=Q&url "=/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum
```

Figure 46. Navigation for Order Now Page

The Order Now Input page is built through the Net.Commerce command `ExecMacro`, which uses the `ordernow.d2w/input` Net.Data macro section. In this macro, the `ordernow.d2w/report` section is used to show the Results Page Accept.

This Net.Data macro also uses the Net.Commerce command `OrderItem Update` to get the ordered item into the database together with the Net.Commerce command `OrderItemDisplay` to show the next page, the Display Current Order page.

3.10 Summary Store Design Considerations

Before you implement an e-business application, you have to complete some planning steps. In this chapter, we discussed general store design considerations. In addition, you should also go through the other planning chapters in this book, for example planning the infrastructure, planning integration with the back-end system, planning payment facilities, and so on.

For the general store design, you have to perform the following planning steps, which can be done in parallel planning rounds:

1. Plan your general e-business behavior and define the appropriate rules for the implementation.

   The summary checklist (see 3.1, “General Considerations” on page 21) can be used to address the major points that have to be considered.
2. Define your product catalog.
3. Plan how to create the image and multimedia files, your logo, and product pictures.
4. Plan how you will work with product descriptions (HTML files).
5. Define how you will implement pricing (discounts) facilities.
6. Define the navigation flow in your e-business store.
7. Define the look and feel (the corporate identity) for the windows that are shown to the customer.
8. Describe the functionality and content of each window.
9. Use frames and perhaps Java Script to enhance usability.
10. Link directly to special categories or products from your home page.
11. Create special search macros.
12. Database design

    Look at the Net.Commerce database. Determine if additional tables or fields are needed. If so, determine if new triggers for this enhancements are needed.
13. Map your navigation flow to Net.Commerce commands.
14. Select, verify, and perhaps change the overridable functions that you need for the Net.Commerce tasks.
15. Establish maintenance procedures.
    a. Use a database cleanup Utility.
    b. Clear log files.
16. Performance issues.
    a. Plan and use caching facilities.
    b. Avoid using the Net.Commerce command `ExecMacro` whenever possible. Use new overridable functions or commands instead.

c. Always insert and update your database tables through the Net.Commerce commands (APIs). Do not use Net.Data macros with SQL statements to do this.

d. Perform back-end program calls asynchronously.

e. Use database cleanup facilities.

f. Use stress test tools.

17. Establish backup and recovery.

- Database backup
- Web pages and macros

Chapter 4. Planning: Language Considerations

When installing Net.Commerce on a system, it is important that you verify your primary language and your system CCSID. The language feature for English is 2924 and has a CCSID of 037.

When you configure the Net.Commerce instance, you need the following information:

- **Language feature** — Refers to which language you have and want to use regarding your Net.Commerce program.

  The other fields regarding language automatically receive the recommended settings based on the language feature code.
- **CCSID** — The CCSID that your system is using.
- **Language ID** — The Net.Commerce instance and the database collection is created based on this value.
- **Locale** — Refers to where the locale object is located.

  A locale is the definition of the subset of a user's environment that depends on language and cultural conventions.
- **Default file system CCSID** — The CCSID that the Web server instance uses, which corresponds to the Net.Commerce instance.
- **Default Net CCSID** — The CCSID that the Web server uses to determine the ASCII to EBCDIC translation that is required for incoming requests and HTTP responses.

If you are planning to install an English version of Net.Commerce, you may need to change some of the parameters to be suitable for your environment. We do not recommend that you change any of these values unless you are an advanced user.

Note: Make sure you have the latest PTFs installed on your AS/400 system.

Chapter 5. Planning: Integration with the Back-End Systems

Integration with back-end or existing applications is the key to developing a successful Net.Commerce site. The integration process usually involves both applications and data.

Net.Commerce provides a high degree of customizing, and has a rich set of overridable functions (OF). These OFs provide the mechanism by which legacy application functions can be incorporated into the Net.Commerce environment. The OFs can also be used to add a new function that may not be present in the legacy applications. The OFs and supporting documentation are described in detail in the Net.Commerce manuals. This chapter provides a high-level overview of how Net.Commerce can be integrated with legacy applications and data.

While using the OFs to customize the Net.Commerce environment is optional, most customers find that integrating legacy data is mandatory. Most customers who have products, goods, or services that they wish to make available in a Net.Commerce environment already have an "inventory" file, and require that information to be used by Net.Commerce.

As described in earlier chapters, Net.Commerce uses its own files, and cannot simply point to existing files. The challenge then becomes how to integrate the existing information or data into the Net.Commerce files.

5.1 Data Mapping

The first step in the data mapping process is to understand the Net.Commerce database schema. This includes the tables, required fields, and primary and foreign keys. This information can be found in the guide *Net.Commerce for AS/400 Commands, Tasks, Overridable Functions, and Database Tables* (dbtodcmd.pdf), which is shipped in softcopy form with the Net.Commerce product. You should review the tables and fields used by Net.Commerce and the information required to fill these fields with meaningful data. You should develop a paper table that maps the Net.Commerce data that you need with the source of the data in your existing back-end system. Refer to 15.3, “Importing Data by Example” on page 330, to see the technique that we used in our sample store.

5.2 Integrating the Data

In addition to the integration of legacy application functions into the Net.Commerce environment, it is important to be able to integrate legacy application data. While some view the task of integrating function, which requires "programming" to be more difficult than integrating data, true integration of data may require more skill. This is due to the dual aspects of integrating data.

First is the task of populating the Net.Commerce database with legacy data. There are several techniques that can be used to accomplish this. Most are reasonably straightforward, if not a bit tedious. Second, and perhaps more importantly, is the task of keeping the Net.Commerce data and the legacy data fully synchronized, so that changes to either set of files is replicated in the other. This task may require a combination of data management skills and programming skills, depending on the particular implementation.

For our design, we elected to develop an RPG program that generates a mass import file. This file can be used to do an initial load and on a periodic basis, to re-sync our Net.Commerce database. In addition, we elected to use triggers to add products as they are added to the back-end system.

5.3 Integrating with Applications

Net.Commerce is a front-end application that provides the tools to quickly set up and populate a mall through which shoppers can browse and place orders. However, shoppers cannot purchase anything unless Net.Commerce is linked to a back-end system that actually fulfills the order and collects the payment. Net.Commerce requires the following back-end systems to complete the shopping process:

- Payment system that verifies and collects credit card payment for goods purchased
- Accounting system that handles inventory, invoicing, and accounts receivable
- Order fulfillment that handles the shipping of purchased goods

Net.Commerce provides macros and overridable functions that can be used to link to these systems. With these links in place, Net.Commerce provides a complete Internet shopping experience from browsing for products to actually purchasing online followed by delivery.

When linking Net.Commerce to existing, applications you can:

- Create new commands — Net.Commerce commands represent a business process such as processing an order.
- Modify or create overridable functions — Overridable functions in Net.Commerce are a precise piece of business logic such as the updating of product inventory.
- Modify or create Net.Data macros.

5.4 Synchronizing the Net.Commerce Database with Back-End Data

Overridable functions (OF) are the recommended way to deal with situations where Net.Commerce changes data arising out of shopping activities, such as inventory or shipping. However, there are certain cases where you have to use some other approach to synchronize the Net.Commerce database with some back-end or legacy system database. Some examples are:

- **To read from the Net.Commerce database into a back-end system.** For example, you may want to synchronize shopper demographics with your data warehouse.
- **When there is no OF available.** For example, a shop selling gambling tickets may have rapidly changing ticket (product) descriptions, depending on the state of the game. There is no OF that deals with product descriptions, so this can be written directly into the database with some process that synchronizes the descriptions with another table.
- **For mass scale updates.** You may want to update all product names in a particular category because your branding has changed.

- **For connectivity or security reasons.** You may not want to link Net.Commerce to your legacy system directly, because of your company's security policies, because your legacy system is on a platform or data format that cannot be linked easily, or because you do not have enough information about your back-end system to code the links. In these and other cases, it may be easier to write a batch update program that periodically updates prices, taxes, inventory, and so on directly into the Net.Commerce database and does not change the behavior of the OF set.
- **For performance.** An OF task may be quite slow because it incurs repeated overheads. It may also improve performance if data, such as price, was directly available in the Net.Commerce database.

The following methods can be used for database synchronization:

- **Mass import**, as discussed in 10.4, “Populating the Net.Commerce Database” on page 170, can be used to update the Net.Commerce database. Some code needs to be written to generate an input file of the required format from the back-end data. Mass import can then be run against this file periodically using the job scheduler.
- **Synchronization tools** such as Data Propagator or Data Mirror. In this case, the user has to consider if the replication is one-way or both-ways: Can the tool handle updates being done at both ends? Also, replicating tools such as data propagator generally updates identical tables or columns. Therefore, it may be necessary to create a large number of definitions if your back-end data does not match Net.Commerce data closely in structure. However, if the definitions are in place, these tools can provide reliable and usually very efficient replication.
- **Custom daemons** that copy data periodically from one set of tables to another. This requires you to invest more heavily in programming and maintenance of code. This approach is useful when there are special requirements such as a stockbroker reading prices from an online information service or cases where only a subset of the data needs to be updated.

Important

We *strongly* recommend that you use overridable functions (not Net.Data macros) to update database information. For performance reasons, the rollback and commit process is controlled in Net.Commerce, not Net.Data. This makes it possible for a Net.Data macro running within Net.Commerce to request a database update, and fail without returning a fail code.

Chapter 6. Planning: Payment Collection

This chapter contains the information you must know to be able to use different payment methods. The major advantage of SET over existing security systems is the addition of digital certificates that associate the cardholder and merchant with their financial institutions and the respective payment brands, for example, MasterCard, Visa, and so on. Digital certificates reinforce existing trusted business relationships and protect against fraud at a level that existing systems do not. For example, SSL provides security in the transmission of sensitive data, but does not guarantee the identity of the parties involved in the transaction

6.1 Secure Electronic Transaction (SET)

SET is an open-network payment-card protocol that provides greater confidentiality, greater transaction integrity, and less opportunity for fraud at all transactions points than any other existing secure payment system. The process involves a series of security checks performed using digital certificates, which are issued to participating purchasers, merchants, banks, and payment brands.

There are five main parties involved in a SET transaction:

- The cardholder
- The merchant
- The issuer (the customers financial institution, which provides the payment card to the customer and the payment to the merchant)
- The acquirer (the merchants financial institution, which enables the merchant to accept a payment card brand and issues the captured payment to the merchant)
- The certificate authority (a trusted third party that can certify the identities of the customer, the merchant and the acquiring institution to each other)

Four of these parties require their own SET software. The issuer communicates with the acquirer over a secure network or other communications channel, and therefore, does not need a secure Internet implementation.

SET has four components:

- A **Cardholder Wallet** component that is run by an online consumer enabling secure payment card transactions over a network. SET Cardholder Wallet components must generate SET protocol messages that can be accepted by SET Merchant, Payment Gateway, and Certificate Authority components.
- A **Merchant Server** (Payment Server) component that is run by an online merchant to process payment card transactions and authorizations. It communicates with the Cardholder Wallet, Payment Gateway, and Certificate Authority components.
- A **Payment Gateway** component that is run by an acquirer or a designated third party that processes merchant authorization and payment messages (including payment instructions from cardholders) and interfaces with private financial networks.
- A **Certificate Authority** component that is run by a certificate authority that is authorized to issue and verify digital certificates as requested by Cardholder

Wallet components, Merchant Server components, or Payment Gateway components over public and private networks.

Some benefits to merchants for implementing SET are:

- Increased sales from existing online shoppers who can now more confidently expand the number of merchant sites where they shop
- Additional sales from consumers who were traditionally constrained from electronic shopping due to their concerns about security on the Internet
- Increased savings through a reduction of exception handling
- Reduced costs associated with fraud

The SET logo or SET mark is a visible symbol signifying that software complies with the SET specification (Figure 47).

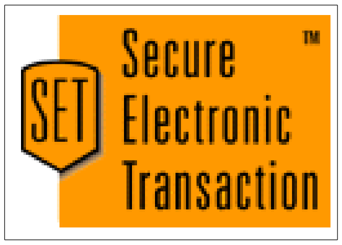


Figure 47. SET Logo

For more information about SET, read the redbook *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*, SG24-4978. You can also refer to the site on the Web at: `http://www.setco.org`

6.1.1 Payment Server

The Payment Server provides payment services on the Internet by taking credit card payments from consumers. The Payment Server runs at a merchant, and is used in conjunction with online shopping software such as Net.Commerce. It supports the SET protocol developed by Visa, Mastercard, IBM, and others. The SET protocol uses strong cryptographic techniques to ensure the transaction data is kept private and not improperly modified.

The Payment Server can obtain credit card approvals and capture funds by communicating with a payment gateway, which runs at a bank (typically called an *acquirer*). In addition, it can process deposits and credits or perform reversals.

Figure 48 on page 93 shows the other servers with which the Payment Server has to interact.

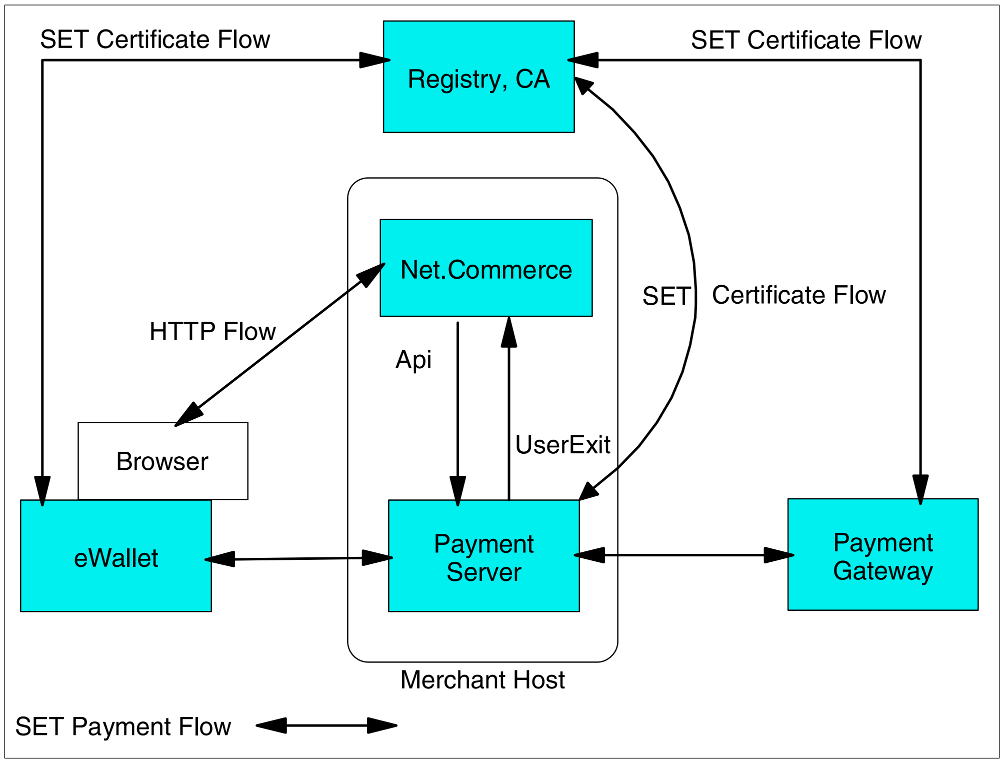


Figure 48. Payment Server

The certificate authorities (CAs) are divided in brands and server type. For more information, see 6.1.3, “SET Certificate” on page 95.

6.1.2 A Payment Server Transaction

The Payment Server interacts with other applications and servers to achieve Secure Electronic Transactions. Also, the Net.Commerce e-business application has commands to work with the Payment Server.

There are two Net.Commerce commands you can use. One works with the eWallet of your customer, as described in 6.1.2.1, “Net.Commerce SET Command” on page 93. The other works without eWallet. See 6.2.1, “Merchant Originated Payment” on page 96, for more information.

6.1.2.1 Net.Commerce SET Command

The Net.Commerce SET command is used to ask the server to send a SET wakeup message to the cardholders browser to launch the wallet application. This command is launched when the cardholder clicks on the Purchase With My Wallet button of the order page. The SET wallet will be displayed and the shopper can pay for the order.

```
http://host_name/cgi-bin/ncommerce3/pay_wakeup
?merchant_rn=merchant_ref_num&order__rn=order_ref_num
```

6.1.2.2 The Payment Flow

The steps following Figure 49 on page 94 describe the process of a typical transaction using the IBM Payment Server.

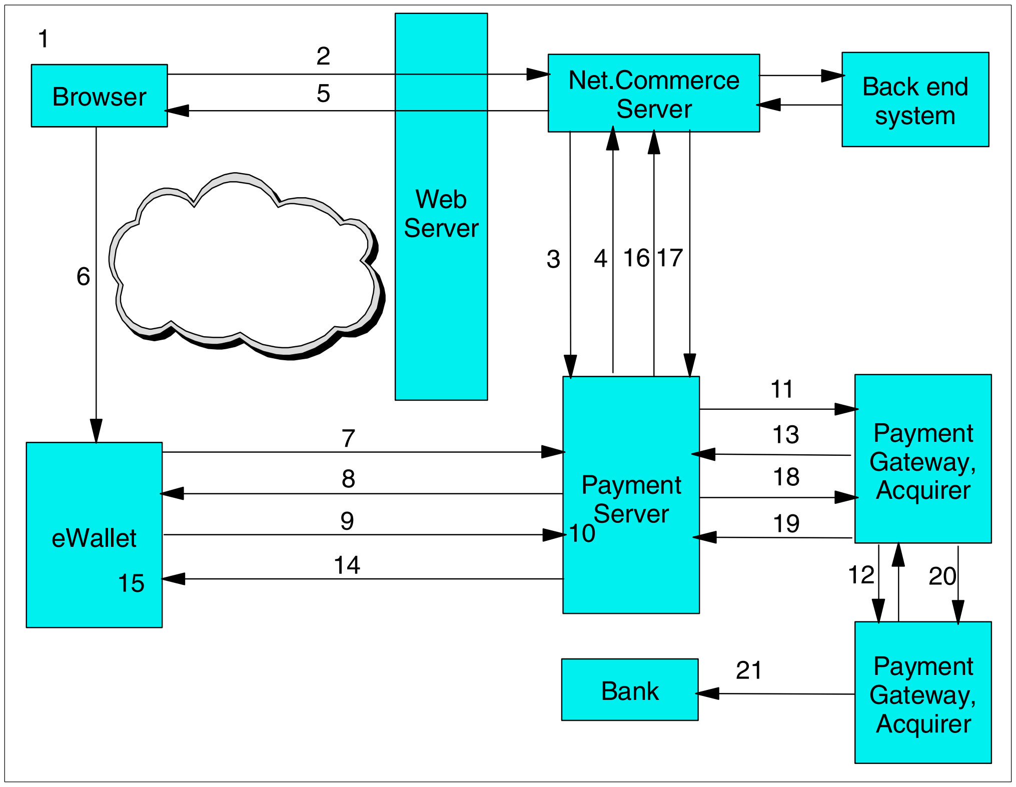


Figure 49. Payment Server Transactions

1. A cardholder decides to make a purchase.
2. When the cardholder clicks the Buy button, a `pay_wakeup` command is sent to the Merchant Server, such as Net.Commerce.
3. The Net.Commerce server calls the Payment Server `etReceivePayment()` API.
4. That command causes the Payment Server to send a payment initiation message, known as a *wakeup message,* back to the Net.Commerce server.
5. The Net.Commerce server passes the message back to the cardholder's browser.
6. This message starts the cardholder's wallet software.
7. The wallet software sends the `PInitReq`, Payment Initialization Request, to the Payment Server.
8. A `PInitRes`, Payment Initialization Response, message is generated by the Payment Server and sent back to the cardholder.
9. The wallet displays the Verify Merchant dialog box to the cardholder, who clicks OK. This causes the wallet software to send the `PReq`, Purchase Request, message to the merchant. This message includes the order information, the payment card information, and the cardholders certificate, all encrypted and digitally signed.
10. The Payment Server checks to see if authorization should be done at this point. For example, if the merchant's acquirer is closed on that day, the process may be delayed until the acquirer is available.

11. When authorization can be done, the server generates an `AuthReq`, Authorization Request, sends it to the acquirer, and waits for `AuthRes`, an Authorization Response message.
12. The acquirer software or Payment Gateway receives the request. Using a normal back-end network or other communication channels, the acquiring institution contacts the cardholders issuing institution. It checks that the payment card is valid and that the cardholder has sufficient funds or credit to make the purchase.
13. The `AuthRes` message is received by the Payment Server and processed. Information is stored in the database for record-keeping and further order processing.
14. A `PRes` or Purchase Response message is generated by the Payment Server and sent to the cardholders wallet application.
15. Assuming the Payment Gateway has confirmed authorization, the `PRes` tells the wallet software that the order has been authorized.
16. The merchant can now fulfill the order.
17. When the goods are shipped, the merchant requests payment.
18. The merchant now begins the capture process by sending a Capture Request to the Payment Gateway. *Capture* is the transfer of funds from the cardholder’s issuing institution to the merchants acquirer and onward to the merchant.
19. The acquirer software receives the capture request and sends it a capture response message.
20. The acquirer uses the closed (back-end) network to contact the cardholders issuing institution and request the transfer of payment.
21. The acquirer deposits the payment to the merchant’s bank account.

6.1.3 SET Certificate

The digital certificates that are used in SET are not the same digital certificates that are used during normal SSL. The Payment Server is not using SSL. The Payment Server performs its own encryption of the payment data in a manner which is much more secure than SSL. The SET process uses special certificates and 128-bit encryption for the credit card information, even outside of North America.

The SET specification requires a hierarchy of trust that is very similar to today’s global payment system model. Cardholders and merchants have trusted relationships with their financial institutions. The financial institutions have existing relationships with one or more payment card brands. Because of the open network environment, SET requires an additional level of trust to authenticate the individual brands. An overall industry entity defined as the SET root certificate authority (CA) authenticates the brands within the SET trust hierarchy.

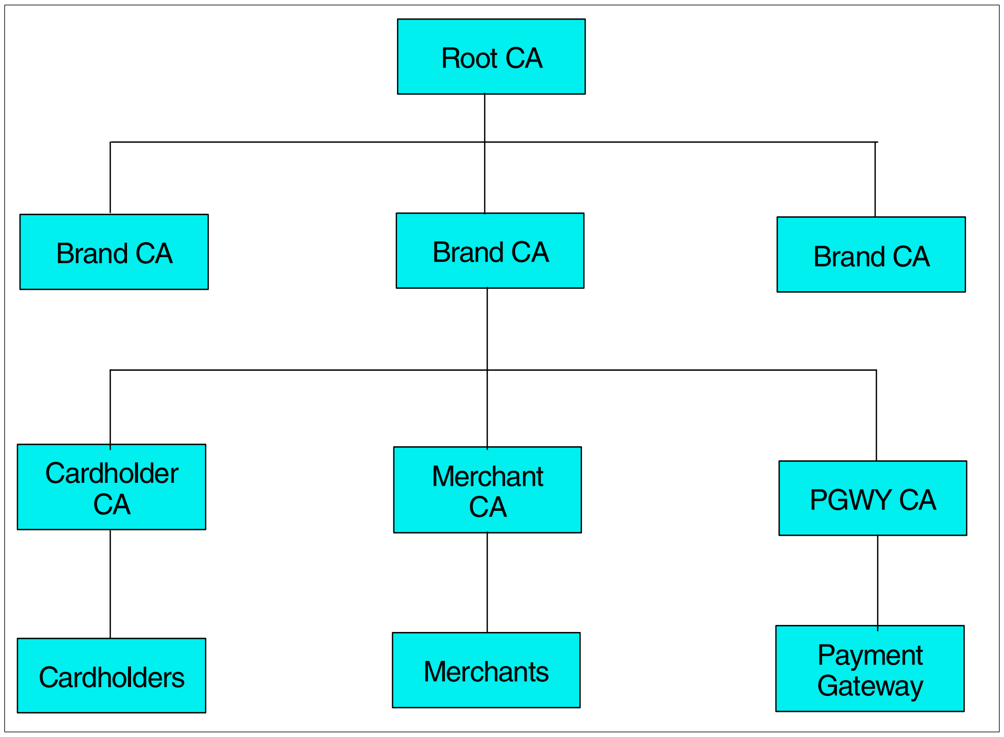


Figure 50. Brand CA

SET represents and verifies each of these trusted relationships through the issuance of digital certificates. The SET root CA issues digital certificates to brands that meet SETCo's brand requirements. A *brand* is an entity that issues payment cards with its own distinct logo. Once a brand issues its certificates, the brand can then sign the certificates and issue them to Cardholder CAs, Merchant CAs, and Payment Gateway CAs.

To use SET as a merchant, you must register with a certificate authority (CA) before you can receive SET payment instructions from cardholders or process SET transactions through a payment gateway. You also need a copy of the registration form from your financial institution. Your software must identify the acquirer to the CA. For the latest information about CA for SET, go to the Web site at: `http://www.setco.org`

6.2 SET without a Wallet

Another scenario to consider is when the cardholder only uses a Web browser and supplies credit card information directly to the merchant. The credit card information is protected by SSL when it flows to the merchant. The Payment Server still communicates with the Payment Gateway using the SET protocol. The IBM solution for that is Merchant Originated Payment.

6.2.1 Merchant Originated Payment

Credit card authorization through the IBM Payment Gateway, also called *Merchant Originated Payment*, provides merchants with an option for shoppers who want security without having to download and install a wallet.

In a typical SET implementation, the merchant customizes order forms to allow shoppers to request the payment Initiation message, also known as the *wakeup message*, from the Net.Commerce server. When the shopper's Web browser receives this payment initiation message, the browser launches a wallet application, such as the IBM Consumer Wallet, which must already be installed on the shopper's machine. Shoppers specify payment card information, and select the payment card to use from the wallet application window.

With Merchant Originated Payment, the wallet is not necessary. Merchant Originated Payment is an IBM extension based on the SET protocol that allows the merchant to receive credit card information through any mechanism, such as over the phone or through the store's online order forms. If the shopper submits credit card information through the store's online order form, when the form is submitted, the credit card information is encrypted using SSL. It is then passed to the acquirer, using regular SET messages, through the IBM Payment Gateway. However, Merchant Originated Payment does not perform cardholder authentication the way that a wallet application does. It is an attractive payment method because shoppers do not need to download and install the wallet software. Note these points:

- From the IBM Payment Gateway point of view, the Merchant Originated Payment is the same as a regular payment transaction, without cardholder authentication. The acquirer must also use the IBM Payment Gateway.
- To support Merchant Originated Payment, the Payment Gateway's encryption certificate must specify `cardCertRequired=FALSE`.

The Net.Commerce command for working with the merchant originated payment is `pay_accept?merchant_rn=$(MerchantRefNum)&order_rn=$(order_rn)`.

You use the Net.Commerce command `pay_accept` instead of the `OrderProcess` command.

6.3 Payment Server Planning Tables

To use the Payment Server, there are some tasks that you must perform. Use Table 6, Table 7 on page 98, Table 8 on page 99, and Table 9 on page 100 when you plan the Payment Server installation.

Table 6. Payment Server Check Table

Information Needed to Install and Configure the Payment Server	**Answer**
Is the information in Table 7, 8, and 9 filled in?	
Is the firewall configuration updated?	
Is the Payment Server installed and created?	

The Payment Server must be installed and created prior to configuration of the Net.Commerce server. Otherwise, you cannot select to use the Payment Server. You have the possibility to install the Payment Server later and then return to the Net.Commerce configuration and change the setting.

You can only use one Net.Commerce instance per Payment Server. However, the Payment Server can support multiple merchants and several brands.

The merchant reference number must be unique. You can have merchants with the same merchants number in different instances.

You need the following information to complete the acquirer configuration on your Payment Server. These fields are shown in Table 7.

Table 7. Acquirer Configuration Planning Worksheet

Field Name	Value	Description
Merchant number		**Required field** — Numeric character string, 1-9 characters, Valid values: 0-999999999: This field can only be chosen from the list of merchants previously configured as payment systems
Account number		**Required field** — Numeric character string: Obtain this value from your acquirer.
Signing brand ID		**Required field** — Text string, 1-40 characters. The Brand ID from the certificate of the signing brand. Note: A merchant may support multiple brands but only one signing brand ID per Acquirer.
SET profile		**Required field** — Integer: The numeric representation for the acquirer profile the merchant is using for batch transactions. Obtain this value initially from your acquirer.
Start time		**Optional field** — Expressed as number of minutes after midnight in the merchant's local time; 0=midnight: Time of day the acquirer opens for business.
Stop Time		**Optional field** — Expressed as number of minutes after midnight in the merchant's local time; 0=midnight: Time of day the acquirer closes for business.
Gateway host name		**Required field** — Text or numeric string, 1-40 characters: Internet host name or IP address of this Payment Gateway.
Gateway port		**Required field** — Numeric character string, default=8888: Port number where this acquirer accepts messages from this particular merchant.
Gateway protocol		**Required field** — HTTP-must be in uppercase: Protocol used by the acquirer (HTTP).
Maximum number of connections		**Required field** — Numeric character string, > or = to 1; default=1: Maximum number of sockets, or connections on the channel.
Read timeout		**Required field** — Number in seconds, 1 - 65535; default=30: Number of seconds to wait for a Payment Gateway to read a SET message from a socket before timing out on a socket. It is also the time between replays.
Number of immediate retries		**Required field** — Numeric character string, 0-65535, default=0: Maximum number of replays in a row before waiting for the interval specified on Delayed retry interval.
Delayed retry interval		**Required field** — Number of minutes, 0-65535; default=0: Number of minutes to wait after the maximum number of replays has been reached before trying again.

Field Name	Value	Description
Confirm delay code		**Optional field** — Numeric character string, default=blank: Set this field to blank unless your acquirer gives you a value.
Confirm delay time		**Optional field** — Number in minutes; default=blank: Set this field to blank unless your acquirer gives you a value.
Pending delay code		**Optional field** — Numeric character string; default=blank: Set this field to blank unless your acquirer gives you a value.
Pending delay time		**Optional field** — Number in minutes; default=blank): Set this field to blank unless your acquirer gives you a value.

You need the following information to complete the acquirer brand configuration on your Payment Server. The fields shown in Table 8 are required.

Table 8. Acquirer Brand Configuration Worksheet

Field Name	Value	Description
Brand ID		**Required field** — Text string 1 to 40 characters long-case sensitive: Payment card brand. Obtain this value from your acquirer.
Acquirer bank ID (BIN)		**Required field** — Numeric string, 1 to 6 characters long: Unique identifier for this acquirer for the brand. Obtain this value from your acquirer.
Acquirer business ID		**Required field string** — 1 to 32 characters long: The business identification number of this acquirer. Obtain this value from your acquirer.
Merchant ID		**Required field** — Numeric character string, 1 to 30 characters long: The SET Merchant ID. Obtain this value from your acquirer.
Have certificate		**Required field** — Default=No: If set to No, the Payment Gateway's certificate is requested from the Acquirer when the Payment Server is started. The Payment Server then changes the setting to Yes.
Terminal ID		**Optional** to SET message, but may be required by acquirer; numeric character string.
Chain number		**Optional** to SET message, but may be required by the acquirer, numeric character string.
Store number		**Optional** to SET message, but may be required by the acquirer, numeric character string.
Agent number		**Optional** to SET message, but may be required by the acquirer, numeric character string.

You need the information in Table 9 to request the Payment Server SET certificate.

Table 9. Payment Server Certificate Planning Worksheet

Field Name	Value	Description
Request URL		**Required field** — Case sensitive; provided by the acquirer.
Financial Institution 125		**Required field** — Questions depending on the CA being used. Review the information you type on this screen. Errors in this information will result in errors on your certificate.
Merchant Name		**Required field** — Questions depending on the CA being used. Review the information you type on this screen. Errors in this information will result in errors on your certificate.
Merchant City		**Required field** — Questions depending on the CA being used. Review the information you type on this screen. Errors in this information will result in errors on your certificate.
Merchant State		**Required field** — Questions depending on the CA being used. Review the information you type on this screen. Errors in this information will result in errors on your certificate.
Merchant Postal Code		**Required field** — Questions depending on the CA being used. Review the information you type on this screen. Errors in this information will result in errors on your certificate.
Merchant Country		**Required field** — Questions depending on the CA being used. Review the information you type on this screen. Errors in this information will result in errors on your certificate.

With the information from the above tables, you are ready to configure your Payment Server.

6.4 Back-End Systems (PO)

You have to decide how you are going to check the authorization and how you are going to request capture. To enable manual capture, authorization and credit transactions, and reversals of payments, you can use the Store Manager. The Store Manager lets you mark the orders you wish to process. Capture is the process by which your acquirer receives the payment from the customer’s financial institution and remits the payment to you.

The following list shows the different alternatives from which you can choose. If you are going to use a back-end system to perform order fulfillment, you may want to request capture from the acquirer automatically when the order is fulfilled. Then, you may select the default value, Auto Auth, and Manual Capture. You have to update the SETSTATUS Net.Commerce table. For more information, see 17.4, “Requesting Capture upon Order Fulfillment” on page 398. These values are defined here:

Auto Auth and Manual Capture (the default)
When the cardholder places the order, the system automatically seeks authorization for the purchase. However, you must initiate capture manually (for example, after the order is fulfilled).

Auto Auth and Auto Capture
When the cardholder places the order, the system automatically seeks

authorization. Upon receiving confirmed authorization, the system then automatically initiates capture. Whether the authorization and capture requests are sent as one message or as two separate messages is determined by the profile used by the acquirer.

Manual Auth and Manual Capture
Authorization and capture are each initiated separately. Capture can be initiated only after authorization has been achieved, at which time the transaction status becomes Capture Ready.

Manual Auth with Auto Capture
Authorization and capture are initiated manually as a single message.

Chapter 7. Planning: Tools to Build the Site

This chapter lists some tools that you may find useful in administrating and customizing the Net.Commerce site on the AS/400 system. Please note that this is only a partial list of tools. You may find it beneficial to use some other tools such as an HTML authoring tool and so on.

7.1 Net.Data SQL Assist Tool

The Net.Data SQL is a useful tool for developing Net.Data macros to be used with Net.Commerce. This section includes information about how to obtain and use the tool.

7.1.1 General Description

The Net.Data SQL Assist is a Java-based GUI SQL statement builder that supports these features:

- An easy-to-use, proven GUI to guide the user through the process of building SQL statements
- Builds SELECT, INSERT, UPDATE, and DELETE statements (including SELECT DISTINCT)
- Allows a user to build multiple conditions (including value lookup, AND/OR, type sensitive entry fields)
- Join tables (inner, right outer, and left outer)
- Select fields to be viewed
- Select sort order
- Allows a user to enter user-defined variables (such as usrvar1) to be used in conditions, values, and so on
- Generates a Net.Data macro file to execute the constructed SQL statement

To run the Net.Data SQL Assist from your PC, you must have the following software:

- Java Development Kit (JDK) or Java Runtime Environment (JRE) 1.1.x. This is available on the Web at: `http://www.javasoft.com`
- Access to the AS/400 Toolbox for Java. For more details, go to the Web site at: `http://www.as400.ibm.com/toolbox/welcome.htm`
- The Net.Data SQL Assist Jar File from the Web site at: `http://www.software.ibm.com/data/net.data/tools/index.html`

7.1.2 Using the Tool

Download the Net.Data SQL Assist tool from the Web site at: `http://www.software.ibm.com/data/net.data/tools/index.html`

Place the tool into a directory on your PC (inst_dir).

The Net.Data SQL Assist is started from the command line and contained in the file {inst_dir}/NetDataAssist.jar.

To start Net.Data SQL Assist with the Java Development Kit (JDK), type:

```
java -classpath %CLASSPATH%;{inst_dir}/NetDataAssist.jar;o:/jt400.zip
NetDataAssist
```

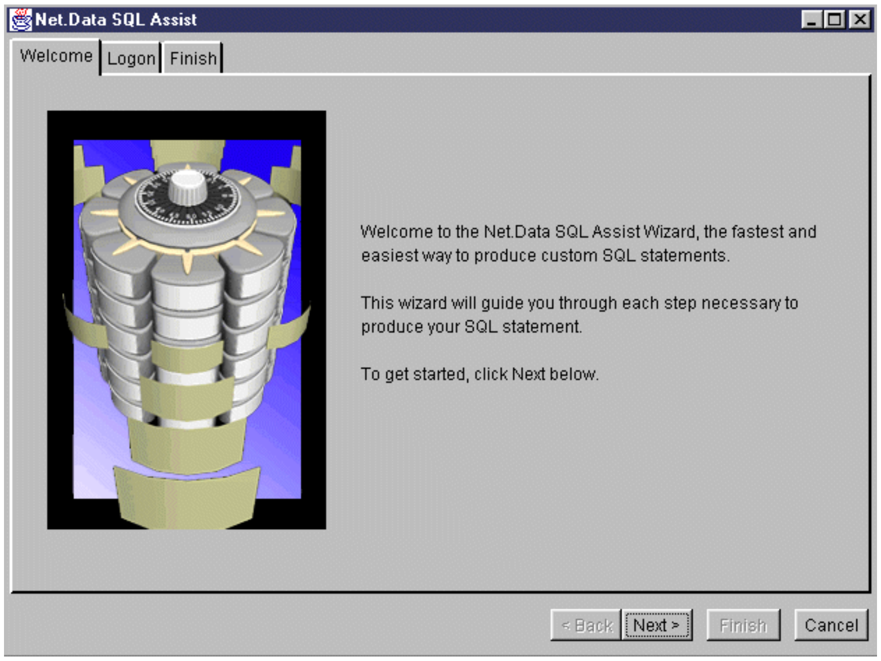
Here, `o:` is the drive mapped to the AS/400 Java Toolbox directory, and inst_dir is the directory that contains the Assist tool.

To start Net.Data SQL Assist with the Java Runtime Environment (JRE), type:

```
jre -cp %CLASSPATH%;{inst_dir}/assist/NetDataAssist.jar;o:/jt400.zip
NetDataAssist
```

Here, `o:` is the drive mapped to the AS/400 Java tool box directory, and inst_dir is the directory that contains the Assist tool.

If your installation is complete, you should see the welcome display shown in Figure 51.



Welcome to the Net.Data SQL Assist Wizard, the fastest and easiest way to produce custom SQL statements.

This wizard will guide you through each step necessary to produce your SQL statement.

To get started, click Next below.

Figure 51. Net.Data SQL Assist Welcome Page

Click on **Next** to proceed to the next window. You receive the logon display as shown in Figure 52 on page 105.

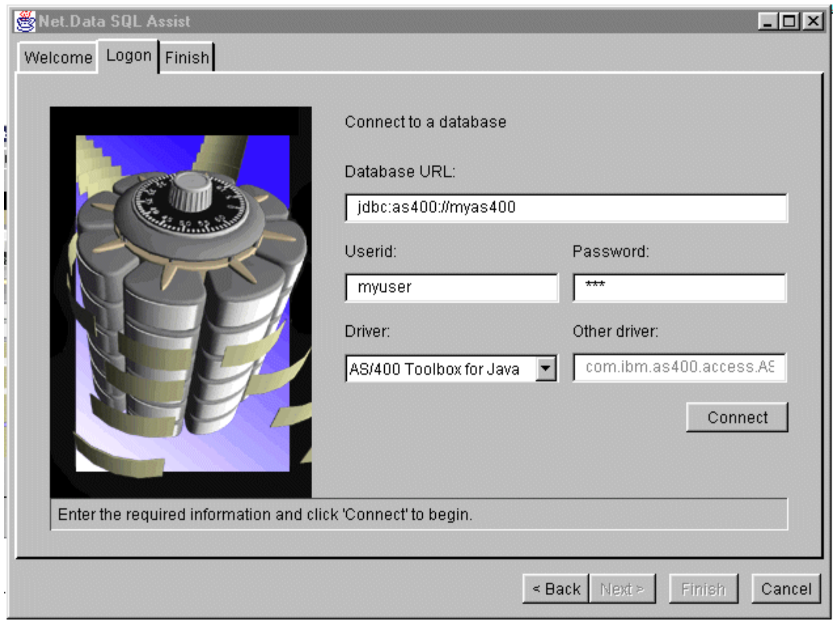


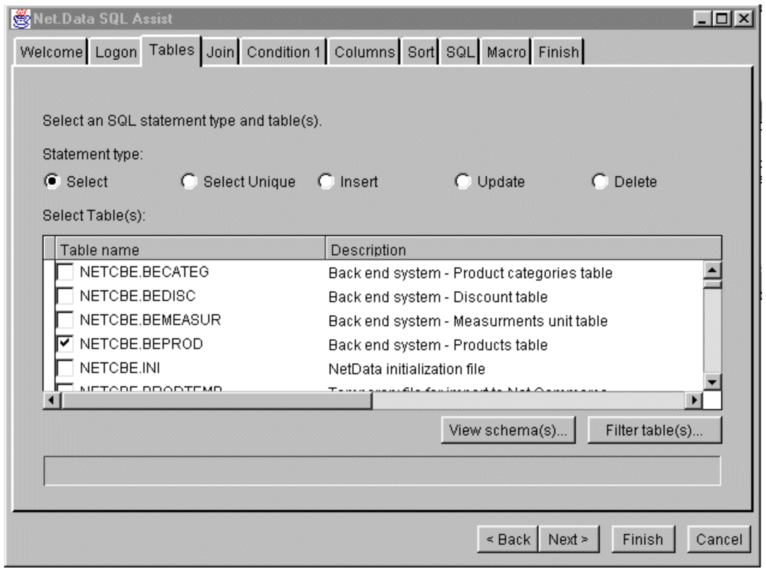
Figure 52. Log on to the AS/400 System

In the window shown in Figure 52, you must connect to your AS/400 system using JDBC. Specify this information:

- *Database URL* must be `jdbc:as400` followed by the AS/400 name or IP address.
- *Userid* is an existing AS/400 user profile.

Driver must be AS/400 Toolbox for Java. The AS/400 toolbox classes must be in the CLASSPATH environment as mentioned in the previous section.

Click **Connect** to connect the AS/400 system. If your AS/400 system was not found or the AS/400 Java Toolbox was not found, a Java error window appears. Upon successful connection, you should see a list of tables in the database as shown in Figure 53 on page 106.



Select an SQL statement type and table(s).

Statement type:

(● Select) ○ Select Unique ○ Insert ○ Update ○ Delete

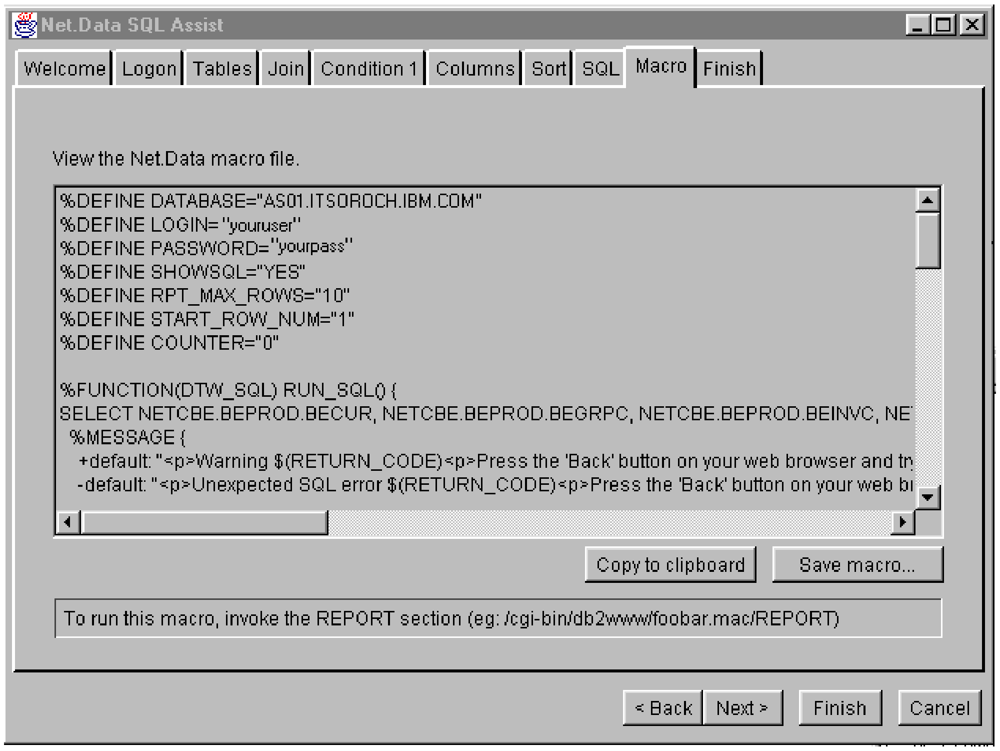
Select Table(s):

Table name	Description
☐ NETCBE.BECATEG	Back end system - Product categories table
☐ NETCBE.BEDISC	Back end system - Discount table
☐ NETCBE.BEMEASUR	Back end system - Measurments unit table
✓ NETCBE.BEPROD	Back end system - Products table
☐ NETCBE.INI	NetData initialization file

Figure 53. List of Tables in Schema

You are now in the main SQL designer. Click **Filter table(s)** to select subset from the table list. Click **View schema(s)** to switch to a different library.

The SQL assistant builds an SQL statement upon your selections. You can use the tool to perform row and column selections, join tables, sort the results, and so on. At the end, a Net.Data macro, which will implement your request, is generated by the tool as shown in Figure 54.



View the Net.Data macro file.

```
%DEFINE DATABASE="AS01.ITSOROCH.IBM.COM"
%DEFINE LOGIN="youruser"
%DEFINE PASSWORD="yourpass"
%DEFINE SHOWSQL="YES"
%DEFINE RPT_MAX_ROWS="10"
%DEFINE START_ROW_NUM="1"
%DEFINE COUNTER="0"

%FUNCTION(DTW_SQL) RUN_SQL() {
SELECT NETCBE.BEPROD.BECUR, NETCBE.BEPROD.BEGRPC, NETCBE.BEPROD.BEINVC, NE
  %MESSAGE {
    +default: "<p>Warning $(RETURN_CODE)<p>Press the 'Back' button on your web browser and tr
    -default: "<p>Unexpected SQL error $(RETURN_CODE)<p>Press the 'Back' button on your web b
```

To run this macro, invoke the REPORT section (eg: /cgi-bin/db2www/foobar.mac/REPORT)

Figure 54. The Net.Data Macro Generated by SQL Assist

You can now save the generated Net.Data macro to your AS/400 system. To do so, you must either map PC drive to the AS/400 IFS system or save the macro to a local directory and then FTP it to the AS/400 system.

We recommend that you save your macro in the root file system and not in the /QSYS.LIB file system. For example, you may save the macro directly to the Net.Commerce macro directory:
`/QIBM/UserData/NetCommerce/instance/<instance_name>/macro.`

To run this macro, invoke the report section from your browser, for example:
`http://www.myas400.com/cgi-bin/db2www/macrofilename/report`

Note

Always test the Net.Data SQL Assist generated macro files against a test database first to ensure that the macro file does what you expect.

7.1.3 Usage Tips

Here are some tips that we found useful when using the Net.Data Assist tool with the AS/400 system:

- Because the Net.Data assist is a general tool, it produces a macro with a few lines that are not needed by the AS/400 system. We recommend, for security reasons, that you manually remove the following lines from the macro:
  - `%DEFINE DATABASE="AS01.ITSOROCH.IBM.COM"`
  - `%DEFINE LOGIN="youruser"`
  - `%DEFINE PASSWORD="yourpass"`
- SQL assist automatically generates macros that retrieve up to ten rows in one SELECT statement, and allows you to scroll to the next and previous 10 rows. You can change the number of rows by adjusting the value of RPT_MAX_ROWS in the macro.
- If you save your macro in a directory or physical file that is not defined to Net.Data, you should modify the db2www.ini file MACRO_PATH statement to allow Net.Data to point to the directory of file. See the Net.Data documentation for further details.
- The SQL assist uses AS/400 Java Toolbox classes to connect to the database. If a low-speed communication link connects your AS/400 system and your PC, the performance of loading the Java Toolbox classes from the AS/400 system may be unacceptable. In this case, having the classes on the PC is a better solution. Copy the jt400.zip file from the directory on the AS/400 system to directory on your local PC. The local directory on the PC must be included in your classpath.

7.2 Entering SQL Statements Using Operations Navigator or SQLUTIL

Net.Commerce works with DB2/400 using the SQL language. All the documentation and samples provided for the product use SQL to interact with the database. Therefore, you must familiarize yourself with SQL syntax, terminology, and tools.

SQL runtime support is part of the operating system. Some other parts of SQL are:

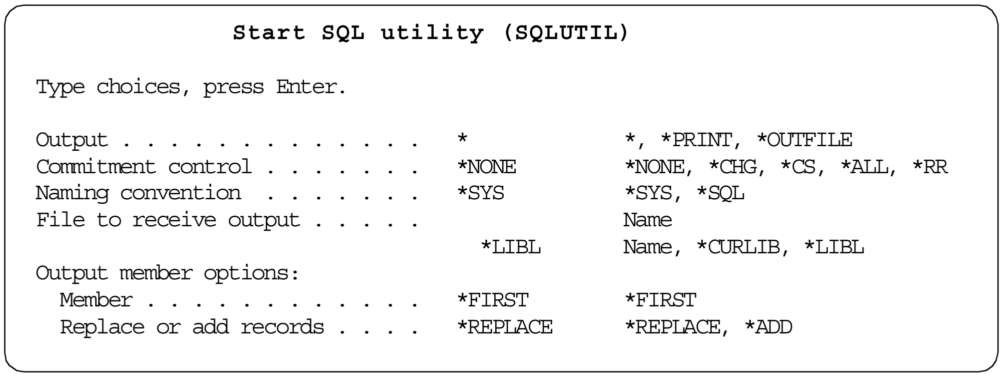
- Licensed program product 5769-ST1, *DB2 Query Manager and SQL DevKit for AS/400,* contains the SQL precompilers, the SQL interactive interface (STRSQL), and the ability to run SQL commands from a source file using the RUNSQLSTM command.
- DB2 Query manager provides a prompt driven interactive interface to SQL statements. The REGOFS and RQSCAP commands listed in Appendix A, “Source Code Samples” on page 463, use the DB2 Query Manager.
- Client Access ODBC driver allows windows applications that are written to MICROSOFT ODBC specifications to transparently access AS/400 database information. For more details on ODBC driver configuration, see Chapter 18, “Generating Net.Commerce Reports” on page 403.
- SQL call-level interface allows users of any ILE language to access SQL functions.

7.2.1 SQLUTIL Command

The SQL Utility (SQLUTIL) command is a "green screen" interface to SQL statements. It is provided "as is" by IBM, along with other useful tools to work with stream files.

The tools are provided through PTF SF49052 and are included in Cumulative Package C8230430. You can also download the tools from the Web site at: `http://service.software.ibm.com/dl/sap/saptools-d`

After following the instructions for creating the tools, the SQLUTIL command is created in library QGPTOOLS. Add the QGPTOOLS library to your library list type `SQLUTIL` and press **F4**. The prompt shown in Figure 55 appears.



```
                 Start SQL utility (SQLUTIL)

 Type choices, press Enter.

 Output . . . . . . . . . . . . .   *             *, *PRINT, *OUTFILE
 Commitment control . . . . . . .   *NONE         *NONE, *CHG, *CS, *ALL, *RR
 Naming convention  . . . . . . .   *SYS          *SYS, *SQL
 File to receive output . . . . .                 Name
                                      *LIBL       Name, *CURLIB, *LIBL
 Output member options:
   Member . . . . . . . . . . . .   *FIRST        *FIRST
   Replace or add records . . . .   *REPLACE      *REPLACE, *ADD
```

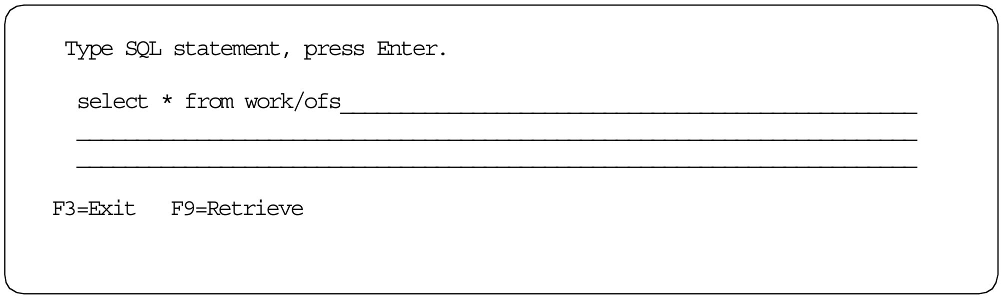
Figure 55. SQLUTIL Prompt

The command parameters are shown in Table 10.

Table 10. SQLUTIL Parameters

Parameter name	Description
Output (OUTPUT)	Specifies whether the output from the command is shown at the requesting workstation, printed with the job's spooled output, or directed to a database file.
Commitment control (COMMIT)	Specifies whether SQL statements are run under commitment control.
Naming convention (NAMING)	Specifies the naming convention used for objects in SQL statements. ***SYS** is the system naming convention and ***SQL** is for SQL naming convention.
File to receive output (OUTFILE)	Specifies the database file that receives the query output.
Output member options (OUTMBR)	Specifies the name of the database member to which the output is directed. The possible action to take values are: ***REPLACE** - The file is cleared before new records are inserted. ***ADD** - New records are added after any existing record.

Press **Enter** in the prompt to display a line screen, as shown in Figure 56, where you can type SQL commands. There is no F4 prompt support for the SQL commands.



```
  Type SQL statement, press Enter.

   select * from work/ofs_______________________________________
   _____________________________________________________________
   _____________________________________________________________

 F3=Exit   F9=Retrieve
```

Figure 56. SQLUTIL Screen

SQLUTIL is a very basic tool. There is no prompt support for the SQL commands. If you perform the select statement, it displays the result using default formatting. However, you may find it useful for ad hoc SQL statements.

7.2.2 Operations Navigator

Operations Navigator is a Windows-like graphical interface included with your system's base software that allows you to point-and-click your way through the AS/400 administration tasks. The installation and configuration of the product can be viewed on the Web at: `http://www.as400.ibm.com/tstudio/opsnav/navframe.htm`

The AS/400 Operations Navigator database administration function is primarily an SQL-based graphical interface to the DB2/400 database. It also allows you to send direct SQL statements to the DB2/400 database.

After Operations Navigator is installed and configured, use the following procedure to send the SQL statement to the database:

1. On the IBM AS/400 Client Access folder, click on **Operations Navigator**.
2. Move the cursor to the AS/400 system with which you wish to work. Click on the **+** symbol to the left of the system name.
3. Scroll down and right-click on **Database**.
4. The menu shown in Figure 57 appears. Click on **Run SQL Statement**.

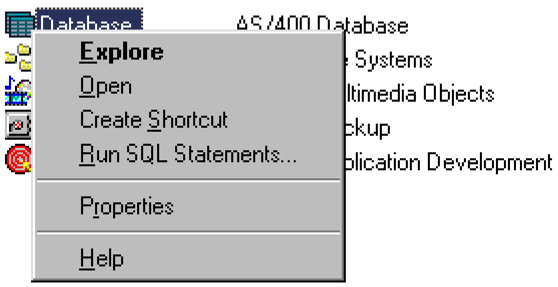
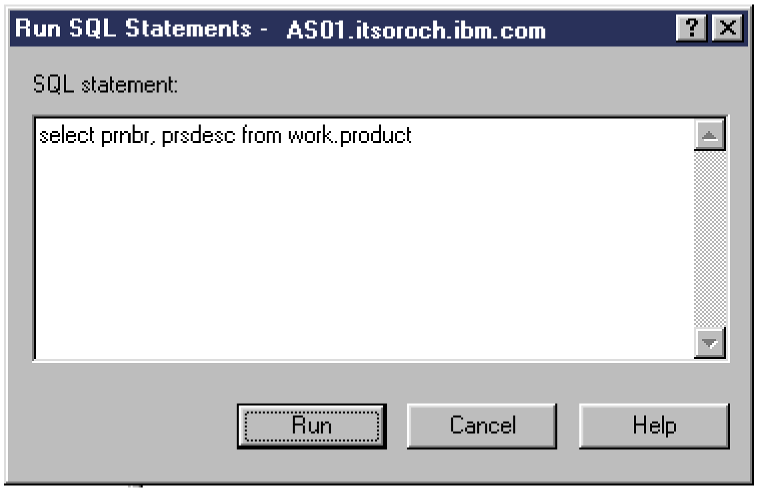


Figure 57. Operations Navigator Pop-Up Menu

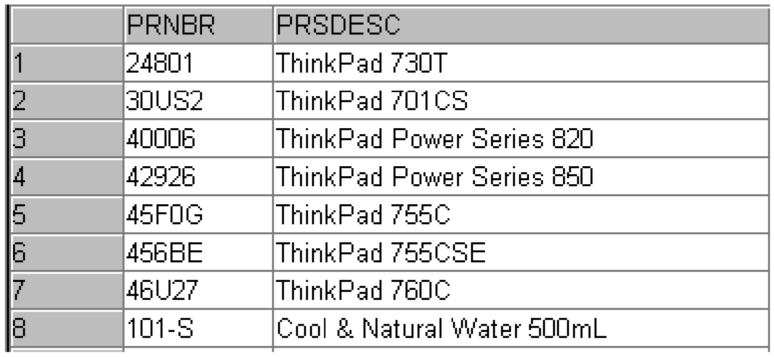
5. Enter any valid SQL statement to run with the DB2/400 database. In our example, we selected the product number and product description from all rows of our product table in our WORK instance (Figure 58).



```
select prnbr, prsdesc from work.product
```

Figure 58. Operation Navigator SQL Interface

6. Click **Run**. The SQL statement executes and the results are returned to your PC screen. Figure 59 on page 111 shows a partial list.



	PRNBR	PRSDESC
1	24801	ThinkPad 730T
2	30US2	ThinkPad 701CS
3	40006	ThinkPad Power Series 820
4	42926	ThinkPad Power Series 850
5	45F0G	ThinkPad 755C
6	456BE	ThinkPad 755CSE
7	46U27	ThinkPad 760C
8	101-S	Cool & Natural Water 500mL

Figure 59. Operations Navigator SQL Result Set from Select Statement

Note: You should only use this SQL interface if you are familiar with the SQL syntax.

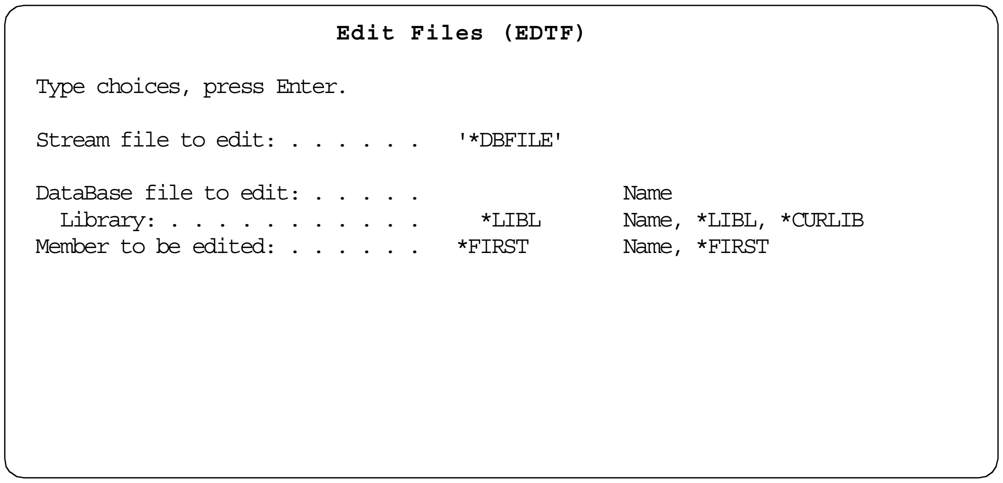
7.3 Stream File Handling Tools

Net.Commerce keeps some of its files in the root file system. For example, the initialization (INI) files for each instance are located in the `/Qibm/UserData/NetCommerce/instance/<instance_name>` directory. IBM provides tools for working with stream files directly from the AS/400 command line. The tools are provided "as is" through PTF (SF49052) and are included in the cumulative package, C8230430. You can also download the tools from the Internet at: `http://service.software.ibm.com/dl/sap/saptools-d`

The current list of tools in the package includes:

DSPSTMF	Display Stream File
EDTF	Edit File
FINDBNDSP	Find Bound Service Program
FINDMODS	Find Modules
MODEXPORTS	List Module Exports
RCLSPACE	Reclaim Space
SQLUTIL	SQL Utility (Described in this chapter)
SAVTOSTMF	Save To Stream File
RSTFRMSTMF	Restore From Stream File
CPYFRMSAVF	Copy From Save File
CPYTOSAVF	Copy To Save File

You may find EDTF to be the most useful tool in the package. After you install the tools, you can test the EDTF command. Type `EDTF` and press **F4**. The command prompt shown on Figure 60 on page 112 appears.



```
                          Edit Files (EDTF)

 Type choices, press Enter.

 Stream file to edit: . . . . . .   '*DBFILE'

 DataBase file to edit: . . . . .                 Name
   Library: . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
 Member to be edited: . . . . . .   *FIRST        Name, *FIRST
```

Figure 60. EDTF Command Prompt

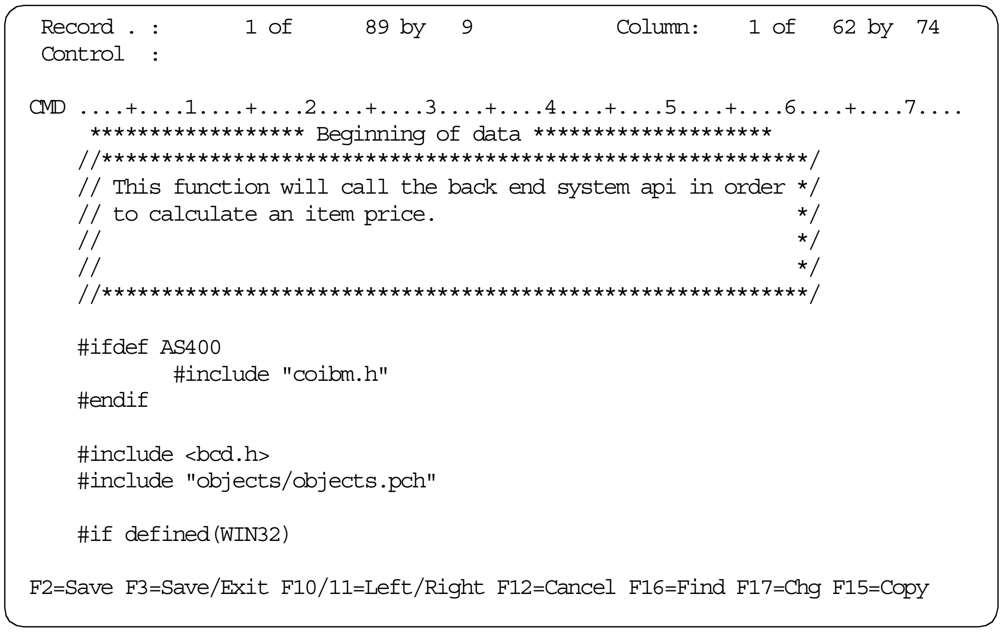
The EDTF Command parameters are shown in Table 11.

Table 11. EDTF Command Parameters

Parameter	Description
Stream file to edit	The full path and name of the stream file you wish to edit. *DBFILE lets us edit database file whose name is written in the database file to edit parameter.
Member to be edited	The member name we wish to edit.

Press the **Enter** key. You will see a screen-line editor, which is similar to an SEU as shown on Figure 61 on page 113. The editor automatically detects the file type (ASCII or EBCDIC). Press **F1** to display a help screen with the available line commands. To activate the command line, press the command to the left of the requested line. For example, press "D" to the the left of a specific line to delete that line.

On the editor screen, press **F3** to exit and save the edited file.

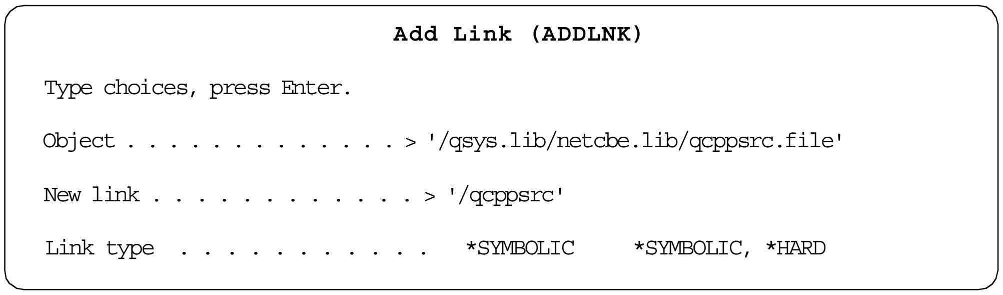


```
 Record . :       1 of      89 by   9           Column:    1 of   62 by  74
 Control  :

CMD ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....
     ****************** Beginning of data *******************
    //**********************************************************/
    // This function will call the back end system api in order */
    // to calculate an item price.                               */
    //                                                           */
    //                                                           */
    //**********************************************************/

    #ifdef AS400
            #include "coibm.h"
    #endif

    #include <bcd.h>
    #include "objects/objects.pch"

    #if defined(WIN32)

F2=Save F3=Save/Exit F10/11=Left/Right F12=Cancel F16=Find F17=Chg F15=Copy
```

Figure 61. EDTF Editing Screen

7.4 Adding a Soft Link to QSYS.LIB Objects

If you are using the CPYTOSTMF or CPYFRMSTMF command, many times you may find it convenient to add a link to your QSYS.LIB files. An example of this technique is shown in Figure 62.



```
                              Add Link (ADDLNK)

 Type choices, press Enter.

 Object . . . . . . . . . . . . . > '/qsys.lib/netcbe.lib/qcppsrc.file'

 New link . . . . . . . . . . . . > '/qcppsrc'

 Link type  . . . . . . . . . . .   *SYMBOLIC     *SYMBOLIC, *HARD
```

Figure 62. Adding Link to QSYS.LIB

After the link is created, the following two statements produce the same result:

```
CPYTOSTMF FROMMBR('/qsys.lib/netcbe.lib/qcppsrc.file/getprice.mbr')
TOSTMF('/usr/test')

CPYTOSTMF FROMMBR('/qcppsrc/getprice.mbr') TOSTMF('/usr/test')
```

Note that the added link allows us to replace the long file name with the equivalent short link name (qcppsrc). This can result in shortened amounts of keying and fewer mistakes.

Chapter 8. Planning: Skills Required for Your Project

The required skills for implementing a Net.Commerce solution are listed here:

- HTML
- IBM HTTP Web Server
- Secure Socket Layer
- Net.Commerce
- Net.Data
- DB2/400
- Payment Server or a SSL Payment applications
- General AS/400 skills
  - CL Programming
  - Working with subsystems
  - Performance tuning
  - Backup and recovery
- Use of Web Browsers

  We used Netscape Navigator for NCADMIN.
- Proficiency at a Programming Language

Optional skills that may be beneficial are:

- C/C++
- Firewall
- Javascript
- Java
- XML
- SQL stored procedures
- TCP/IP configuration and use

Part 2. Implementing the Net.Commerce Site

The second part of this book takes you through the various stages of implementing a Net.Commerce site, including setup, installation, configuration, and more. Plus, it includes three appendices that contain source code samples, additional performance notes, and a problem and solution guide.

Chapter 9. Setting Up SSL Using DCM

To use Secure Sockets Layer (SSL) protocol, you need a digital certificate assigned by a certificate signer, for example, IBM World Registry or VeriSign. To do that, you need to have Digital Certificate Manager (DCM) installed. For more information about the SSL, read *TCP/IP Tutorial Technical Overview*, GG24-3376.

9.1 Transaction Security and Secure Sockets Layer

Transaction security includes several basic elements, such as:

- Confidentiality and privacy
- Integrity
- Authentication
- Accountability

SSL is the *Secure Sockets Layer* protocol defined by Netscape Communications Corporation. It provides a private channel between a client and a server that ensures privacy of data, authentication of session partners, and message integrity.

Digital certificates are used for session partner authentication. Server authentication is common. Client authentication is not yet common, but it is growing in popularity. Keys are the base for end-to-end information encryption.

Confidentiality means that the contents of messages remain private as they pass through the Internet. Without confidentiality, your computer broadcasts messages to the network, which is similar to shouting the information across a crowded room. *Encryption* ensures confidentiality.

Integrity means that the messages are not altered while being transmitted. Any router along the way can insert or delete text, or garble the message as it passes by. Without integrity, you have no guarantee that the message sent matches the message received. Encryption and a *digital signature* ensure integrity.

Authenticity means that you know who you are talking to and that you trust that person. Without authenticity, you have no way to be sure that anyone is who they say they are. *Authentication* ensures authenticity.

There are two ways in which the server uses authentication:

- Digital signature
- Digital certificates

A digital signature ensures accountability. But how do you know if the person sending you a message is who they say they are?

Look at the sender's *digital certificate*. A public key certificate is issued by a trusted third party, known as the *certifying authority* (CA). A browser and server exchange information, including their public key certificate. SSL uses the information to identify and authenticate the sender of the certificate.

A digital certificate is like a credit card with your picture on it and a picture of the bank president with his arm around you. A merchant trusts you more because you look like the picture on the credit card, and they know the bank president trusts you, too.

You base your trust for the authenticity of the sender on whether you trust the third party (a person or agency) that certified the sender. The third party, or *certification authority (CA)*, issues digital certificates.

Trusted third parties verify that the server really is who it claims to be. This verification is provided with a digital certificate (the digital equivalent of your doctor's diploma hanging on the wall). You base your trust for the authenticity of the server on whether you trust the third party that certified the server (the school that issued the diploma). That third party is called a certifying authority (CA).

The term *trusted root* is given to a trusted certifying authority (CA) on your server. A *trusted root key* is the key belonging to the CA.

Authentication can be used server-to-client (server authentication) or client-to-server (client authentication). Server authentication is described earlier. The clients authenticate the servers. With client authentication, the client is authenticated by the server.

Accountability means that both the sender and receiver agree that the exchange took place. Without accountability, the addressee can easily say that the message never arrived. Digital signatures ensure accountability. Accountability is *not* part of the SSL protocol.

9.2 HTTP Server over SSL (HTTPS)

SSL ensures that data transferred between a client and a server remains private. It allows the client to authenticate the identity of the server. In addition, SSL V3 allows a server to authenticate a client.

Figure 63 shows the high-level view of the flow that takes place when a client (browser) sends an HTTPS request to an HTTP server.

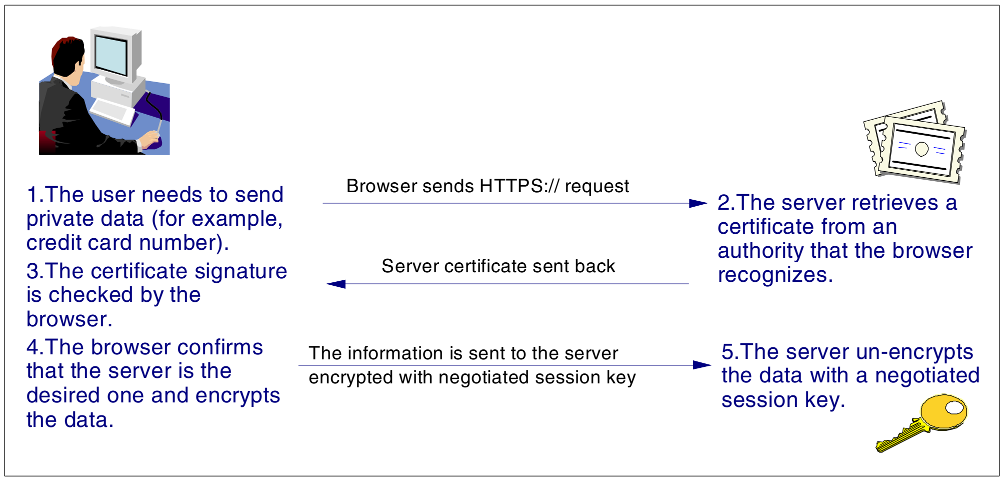

1.The user needs to send private data (for example, credit card number).
Browser sends HTTPS:// request
2.The server retrieves a certificate from an authority that the browser recognizes.
Server certificate sent back
3.The certificate signature is checked by the browser.
4.The browser confirms that the server is the desired one and encrypts the data.
The information is sent to the server encrypted with negotiated session key
5.The server un-encrypts the data with a negotiated session key.

Figure 63. HTTP Server Using SSL

Once your server has a digital certificate, SSL-enabled browsers can communicate securely with your server using SSL. With SSL, you can easily establish a security-enabled Web site on the Internet or on your corporate network.

The benefits of HTTP using SSL include:

- Target server is verified for authenticity
- Information is encrypted for privacy
- Data is checked for transmission integrity

Because HTTPS (HTTP + SSL) and HTTP are different protocols, and usually use different ports (443 and 80, respectively), you can run both secure and non-secure servers at the same time. As a result, you can choose to provide information to all users using no security, and specify certain information only to browsers who make secure requests. This is how a retail company on the Internet can allow users to look through merchandise without security, complete order forms, and send their credit card numbers using SSL security. A browser that does not have support for HTTP over SSL naturally cannot request URLs using HTTPS. The non-SSL browsers do not allow users to send forms that need to be submitted securely.

9.3 Digital Certificates and Certificate Authority

A digital certificate identifies a user or a system and is required before SSL can be used. Once a server has a digital certificate, SSL-enabled browsers, such as Netscape Navigator, can communicate securely with the server using SSL.

A digital certificate is issued by a certificate authority (CA). CAs are entities that are trusted to properly issue certificates and have controls in place to prevent fraudulent use. If you can trust a CA, you can be reasonably certain that any certificate they issue properly represents the individual that is holding it.

Note: The certificate authority charges a fee for issuing a certificate.

Some examples of universally recognized Internet certificate authorities (CA) include:

- Thawte
- VeriSign
- US Postal Service
- AT&T
- MCI

For testing purposes, or for applications that will be used exclusively in an intranet environment, you may issue digital certificates using an intranet certificate authority. The AS/400 system with Digital Certificate Manager (DCM) can act as an intranet certificate authority.

For secure communications, the receiver must trust the CA that issued the certificate, whether the receiver is a browser or a server. Any time a sender signs a message, the receiver must have the corresponding CA certificate and public key designated as the trusted root key.

9.4 AS/400 Implementation of Digital Certificate Management

You can configure your AS/400 system as an intranet certificate authority. Digital Certificate Manager (DCM) is a Web-browser based administration facility that allows you to create, manage, and use certificates within an enterprise and with partners of an enterprise. You can use DCM to request digital certificates from Internet Certificate Authorities such as VeriSign and Thawte. To use all the options available in DCM, you must have *SECOFR and *SECADM authority.

To access the Digital Certificate Manager, click on the hyperlink for **Digital Certificate Manager** from the AS/400 Tasks page. When using Digital Certificate Manager, you can click the **Help** button on any page, at any time, to access online help.

9.4.1 Configuring a Digital Certificate Environment

You can use your AS/400 system to configure a digital certificate environment. You can also configure the HTTP server to use digital certificates and run over SSL.

Follow this series of steps for configuring an intranet digital certificate environment that uses the AS/400 system as a certificate authority:

1. Use DCM to create an intranet CA in one or more AS/400 systems.
2. Using DCM, the intranet CA issues server certificates that can be used on the local server (the same AS/400 system where the CA is configured), or exported to a remote server.
3. For the clients to recognize and trust the server certificates issued by the intranet CA, install the CA certificate in the browsers and designate it as a trusted root.
4. If the server requests client certificates for client authentication, the users must request and install client certificates in their browsers.
5. Configure the HTTP server to enable SSL (SSL On) and specify the key-ring file where the server certificate is stored (keyfile). To optionally authenticate client certificates (SSL_ClientAuth client), add PROTECTION/PROTECT directives to protect resources.

9.5 Creating a Self-signed Certificate

This section describes how to create a self-signed certificate using your AS/400 system as an intranet certificate authority. The steps used in V4R3 and V4R4 of DCM are very similar. Watch for different procedures for the different releases. To test your Net.Commerce site before you deploy it, you have to create a self-signed certificate.

Because self-signed certificates are not recognized by a visitor's browser as coming from a trusted third party, they should not be used in customer transaction situations over the Internet. Use them only on your test and development systems, and for demonstration purposes. You can also use a self-signed certificate for intranet applications.

To obtain a self-signed certificate, perform the following tasks:

1. Create an intranet certificate authority.
2. Create a server certificate with your intranet CA.
3. Configure your HTTP server to use the server certificate.

9.5.1 Creating an Intranet Certificate Authority

Digital Certificate Manager (DCM) allows you to create your own intranet CA on your AS/400 system and use it to issue server and client certificates for testing purposes or applications within your organization. This section outlines the steps you must perform to create a CA on your AS/400 system. You only need to perform this task if the system administrator has not previously created an intranet certificate authority, and if you want to use your AS/400 system to issue intranet server certificates. We recommend that you always create a CA on your AS/400 system in case you need one for testing.

To create an intranet CA on your AS/400 system, follow these steps:

1. Start the HTTP *ADMIN server on your AS/400 system. From the command line, type:

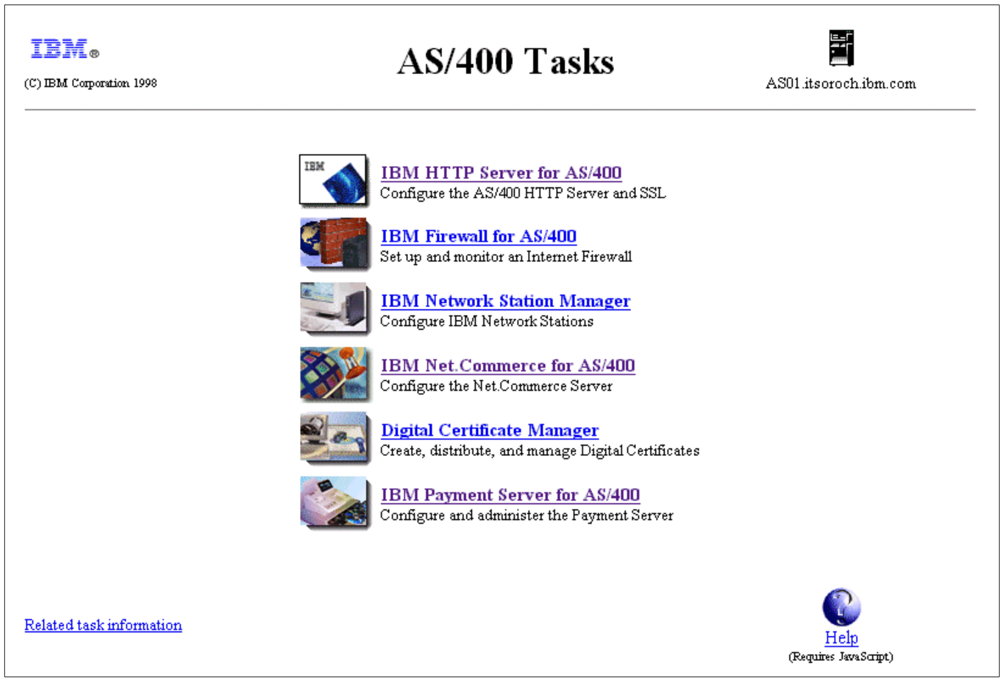
   ```
   STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
   ```

2. Access the AS/400 Tasks page from your browser by entering the URL:

   `http://System_name:2001`

3. You are prompted to enter a user name and password. Sign on with a user that has *SECOFR and *SECADM authority.

   The AS/400 Tasks Page appears as shown in Figure 64.



IBM®
(C) IBM Corporation 1998

AS/400 Tasks

AS01.itsoroch.ibm.com

IBM HTTP Server for AS/400
Configure the AS/400 HTTP Server and SSL

IBM Firewall for AS/400
Set up and monitor an Internet Firewall

IBM Network Station Manager
Configure IBM Network Stations

IBM Net.Commerce for AS/400
Configure the Net.Commerce Server

Digital Certificate Manager
Create, distribute, and manage Digital Certificates

IBM Payment Server for AS/400
Configure and administer the Payment Server

Related task information

Help
(Requires JavaScript)

Figure 64. AS/400 Tasks Page

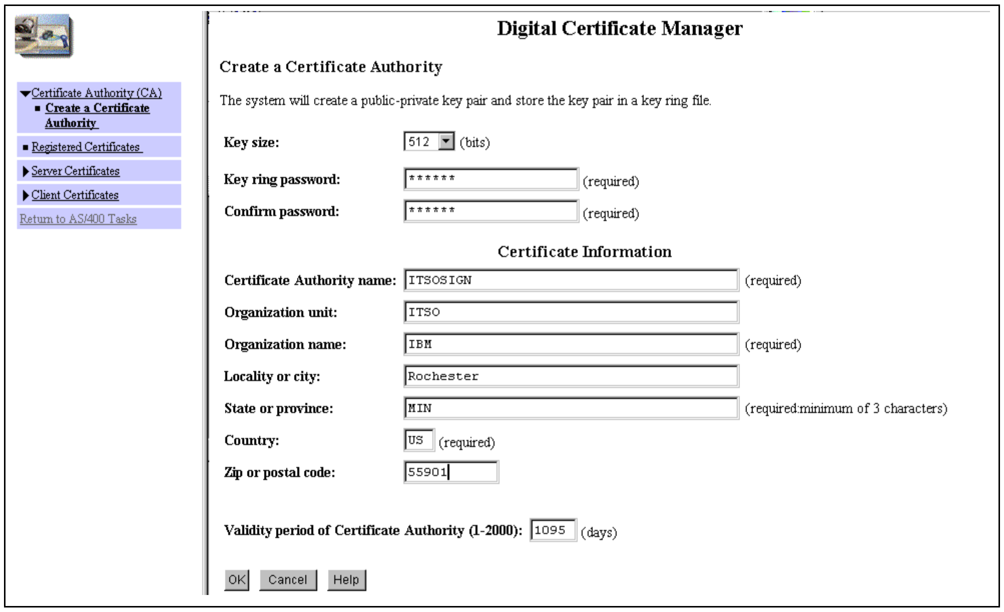
4. Click **Digital Certificate Manager**.
5. Click **Certificate Authority (CA)**.

6. Click **Create a Certificate Authority**.

Note

If a certificate authority (CA) was previously created on your system, the Create a Certificate Authority link does not appear.

7. Complete the Create a Certificate Authority form as shown in Figure 65. Replace the field values appropriately with your organization's information.



Digital Certificate Manager

Create a Certificate Authority

The system will create a public-private key pair and store the key pair in a key ring file.

Key size:	512	(bits)
Key ring password:	******	(required)
Confirm password:	******	(required)

Certificate Information

Certificate Authority name:	ITSOSIGN	(required)
Organization unit:	ITSO	
Organization name:	IBM	(required)
Locality or city:	Rochester	
State or province:	MIN	(required:minimum of 3 characters)
Country:	US	(required)
Zip or postal code:	55901	

Validity period of Certificate Authority (1-2000): 1095 (days)

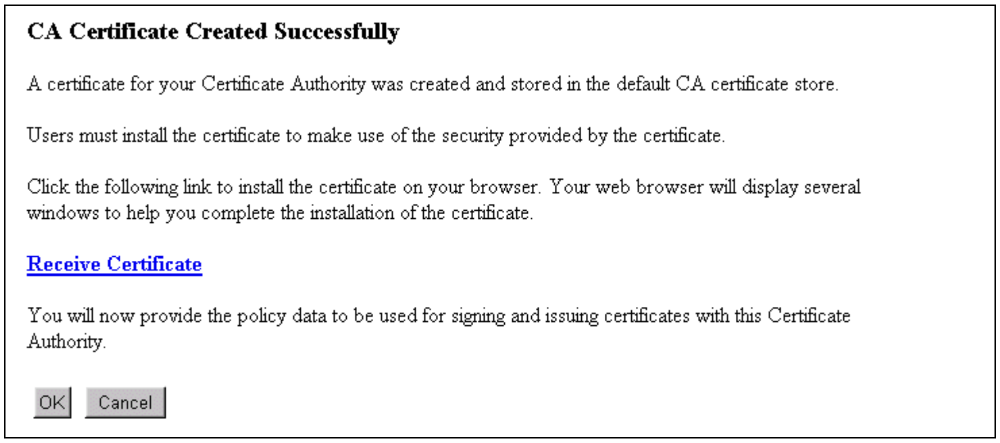
Figure 65. Creating an Intranet Certificate Authority

Click **OK**.

8. After DCM processes the form, it stores a copy of the CA certificate and other information in the following IFS directory:

/QIBM/USERDATA/ICSS/CERT/CERTAUTH/

At this point, you can install the CA certificate on your browser so that it recognizes the certificates issued by the intranet CA. DCM displays a page similar to the one shown in Figure 66 on page 125. The contents of the page vary based on the release.



CA Certificate Created Successfully

A certificate for your Certificate Authority was created and stored in the default CA certificate store.

Users must install the certificate to make use of the security provided by the certificate.

Click the following link to install the certificate on your browser. Your web browser will display several windows to help you complete the installation of the certificate.

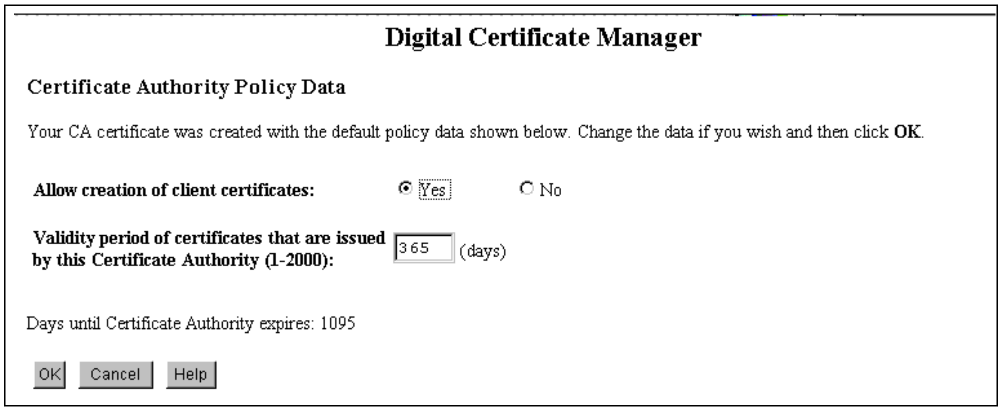
Receive Certificate

You will now provide the policy data to be used for signing and issuing certificates with this Certificate Authority.

Figure 66. CA Certificate Created Successfully

Click **Receive Certificate** if you want to install the CA certificate on your browser now. Or, click **OK** to proceed to the next setup window, and install the CA certificate on your browser at a later time.

9. Complete the CA Policy Data form to set the client certificate policy for your CA. See Figure 67.



Digital Certificate Manager

Certificate Authority Policy Data

Your CA certificate was created with the default policy data shown below. Change the data if you wish and then click **OK**.

Allow creation of client certificates: [✓ Yes] No

Validity period of certificates that are issued by this Certificate Authority (1-2000): 365 (days)

Days until Certificate Authority expires: 1095

Figure 67. Certificate Authority Policy

This is where you define whether your CA can issue and sign client certificates. If the CA can issue client certificates, indicate the length of time for which the certificates will be valid.

10. The following message appears:

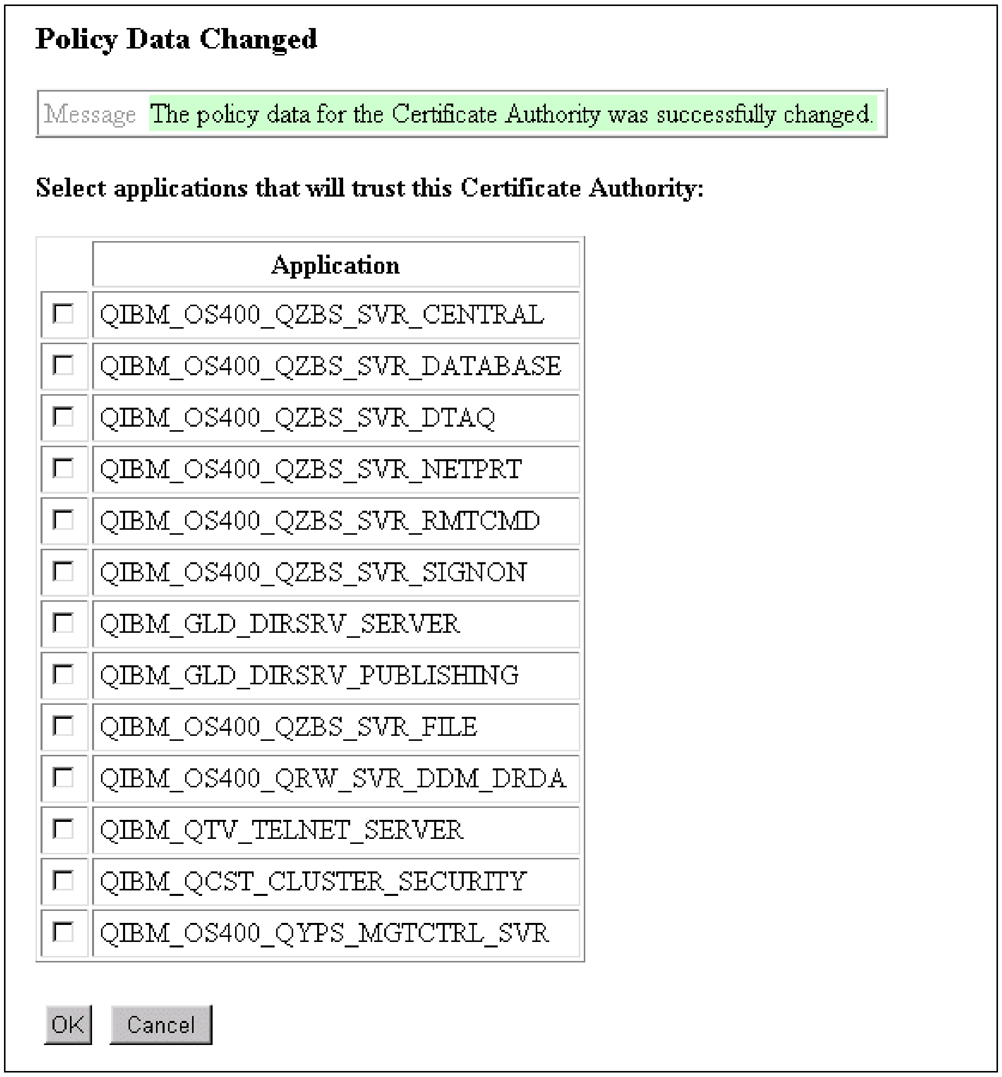
```
The policy data for the Certificate Authority was successfully changed.
```

At this point, you can continue to create a server certificate signed by your certificate authority. This allows server authentication by clients that use this system as a server.

If you are using V4R3, skip to 9.5.2, “Creating a Server Certificate with Your Intranet CA (V4R3)” on page 126.

If you are using V4R4 of DCM, you are presented with a window that allows you to trust this CA for applications. A sample is shown in Figure 68 on page 126. On this panel, select any applications that are going to use this CA for security. If you installed and configured SSL in the HTTP instance used by

Net.Commerce, an entry is listed for the HTTP server instance name. The entry is in the form `QIBM_HTTP_SERVER_instancename`, where `instancename` is the name of the HTTP server instance.



Policy Data Changed

Message The policy data for the Certificate Authority was successfully changed.

Select applications that will trust this Certificate Authority:

	Application
☐	QIBM_OS400_QZBS_SVR_CENTRAL
☐	QIBM_OS400_QZBS_SVR_DATABASE
☐	QIBM_OS400_QZBS_SVR_DTAQ
☐	QIBM_OS400_QZBS_SVR_NETPRT
☐	QIBM_OS400_QZBS_SVR_RMTCMD
☐	QIBM_OS400_QZBS_SVR_SIGNON
☐	QIBM_GLD_DIRSRV_SERVER
☐	QIBM_GLD_DIRSRV_PUBLISHING
☐	QIBM_OS400_QZBS_SVR_FILE
☐	QIBM_OS400_QRW_SVR_DDM_DRDA
☐	QIBM_QTV_TELNET_SERVER
☐	QIBM_QCST_CLUSTER_SECURITY
☐	QIBM_OS400_QYPS_MGTCTRL_SVR

Figure 68. Trusting the CA for Applications

11.After you select any applications, click **OK**. You will receive a message indicating that the system will now create a system certificate.

If you are using V4R4, go to 9.5.3, "Creating a System Certificate with Your Intranet CA (V4R4)" on page 129.

9.5.2 Creating a Server Certificate with Your Intranet CA (V4R3)

After creating the intranet CA, DCM prompts you to create a server certificate. To use Secure Sockets Layer (SSL) for secure Web serving, your server must have a digital certificate. When you create a server certificate in DCM, the server certificate and keys are stored in the following default directory and file:

```
/QIBM/USERDATA/ICSS/CERT/SERVER/DEFAULT.KYR
```

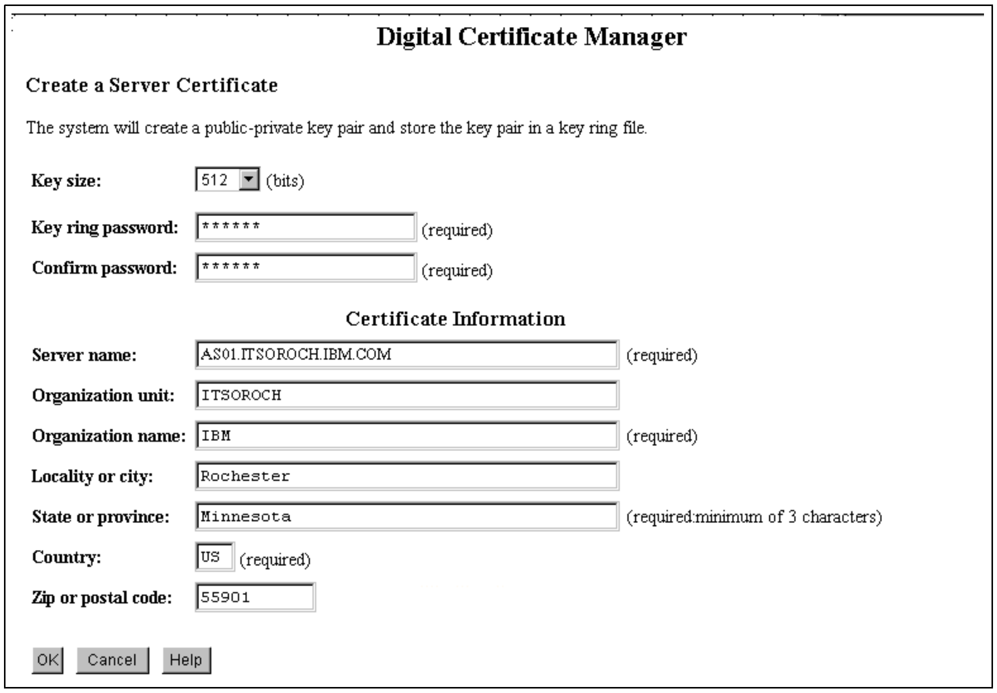
Note

When you create a server certificate, Digital Certificate Manager (DCM) stores a copy of the CA certificate in the server key ring and designates it as a trusted root.

To create a server certificate with your intranet CA, complete the following steps:

1. Complete the Create a Server Certificate form as shown in Figure 69. Replace the field values with your organization's information.

   The options for the key size are determined by the IBM Cryptographic Access Provider (5769-ACx) licensed program product installed on your system. This is the key size that is used to generate your public and private keys.



Digital Certificate Manager

Create a Server Certificate

The system will create a public-private key pair and store the key pair in a key ring file.

Key size:	512 (bits)
Key ring password:	****** (required)
Confirm password:	****** (required)

Certificate Information

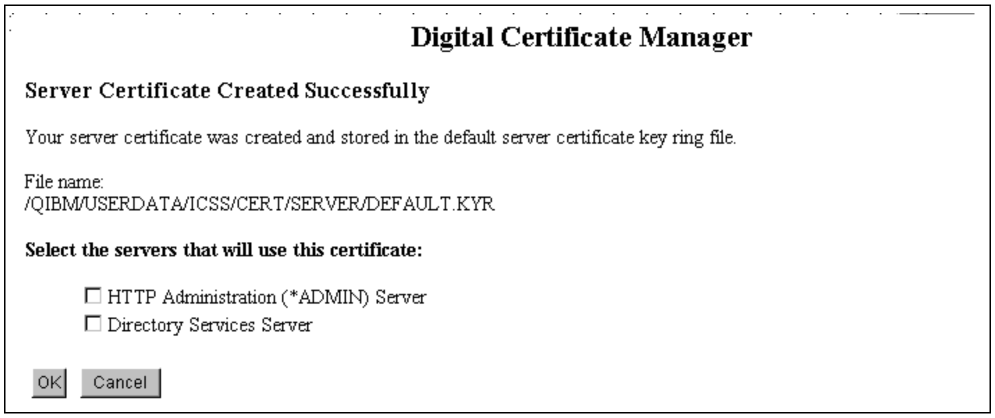
Server name:	AS01.ITSOROCH.IBM.COM (required)
Organization unit:	ITSOROCH
Organization name:	IBM (required)
Locality or city:	Rochester
State or province:	Minnesota (required:minimum of 3 characters)
Country:	US (required)
Zip or postal code:	55901

Figure 69. Create a Server Certificate Page

   By default, the system inserts the fully qualified name of the AS/400 system into the system name field. You can give the server any name. However, the fully qualified TCP/IP host name is usually used for the server name. Some CAs require that the state name be spelled out completely. We recommend that you always use the entire name rather than a short form.

2. Click **OK**.

   The Server Certificate Created Successfully page appears as shown in Figure 70 on page 128.



Digital Certificate Manager

Server Certificate Created Successfully

Your server certificate was created and stored in the default server certificate key ring file.

File name:
/QIBM/USERDATA/ICSS/CERT/SERVER/DEFAULT.KYR

Select the servers that will use this certificate:

☐ HTTP Administration (*ADMIN) Server
☐ Directory Services Server

Figure 70. Server Certificate Created Successfully Page

From this page, you can select whether the HTTP ADMIN server or the Directory Services server (LDAP) uses this server certificate for SSL connections. Do *not* select any of these options.

3. Copy the following file and path name where the server certificate is stored to the clipboard:

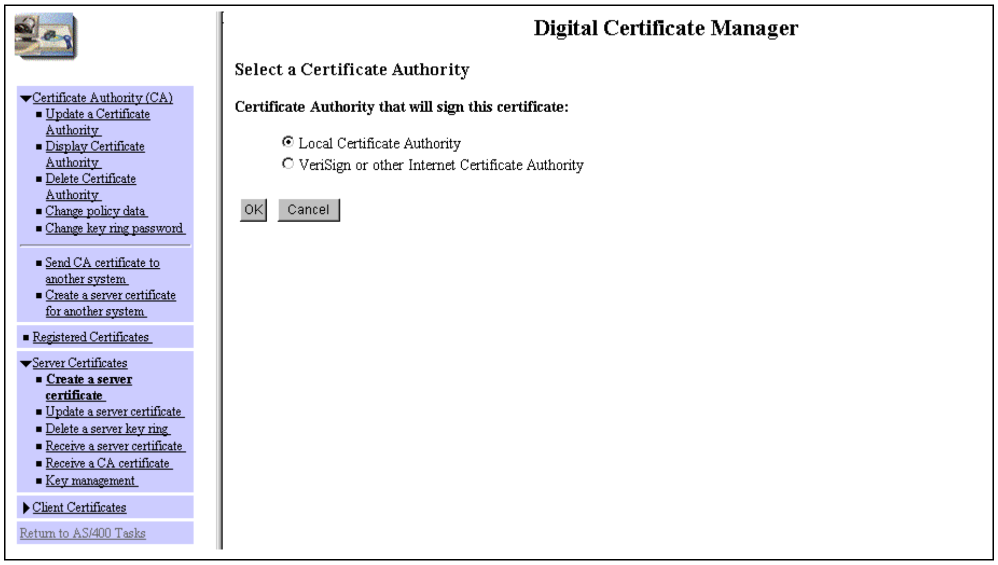
```
/QIBM/USERDATA/ICSS/CERT/SERVER/DEFAULT.KYR
```

Click **OK**. Click **Done**.

9.5.2.1 Creating a Server Certificate with an Existing Intranet CA

The steps to create a server certificate described in the previous section assume that you are creating the intranet CA for the first time. If your administrator has already created an intranet CA and server certificate, you can use the existing server certificate in your HTTP server configuration.

If you want to create a new server certificate using an existing intranet CA, start by clicking **Create a server certificate** under Server Certificates in DCM (Figure 71).



Certificate Authority (CA)
Update a Certificate Authority
Display Certificate Authority
Delete Certificate Authority
Change policy data
Change key ring password
Send CA certificate to another system
Create a server certificate for another system
Registered Certificates
Server Certificates
Create a server certificate
Update a server certificate
Delete a server key ring
Receive a server certificate
Receive a CA certificate
Key management
Client Certificates
Return to AS/400 Tasks

Digital Certificate Manager

Select a Certificate Authority

Certificate Authority that will sign this certificate:

(●) Local Certificate Authority
( ) VeriSign or other Internet Certificate Authority

Figure 71. Creating a Server Certificate with an Existing Intranet CA

Select **Local Certificate Authority**, and click **OK**.

The Create Server Certificate page appears next as shown in Figure 69 on page 127.

9.5.2.2 Authorizing QTMHHTTP to the Key Ring File

You may need to give QTMHHTTP (or the user profile under which your HTTP server runs) authority to the key ring and stash files. The key ring and stash files are created with *PUBLIC authority *EXCLUDE. QTMHHTTP (or the user profile under which the HTTP server runs) must at least have read rights to those files.

Perform the following steps:

1. To authorize QTMHHTTP to the key ring and stash file, type the command:

   ```
   WRKLNK '/QIBM/UserData/ICSS/Cert/Server'
   ```

2. Enter `5` (Next level) to display the files in the directory.
3. Enter `9` (Work with authority) by the key ring file (DEFAULT.KYR).
4. Enter `1` (Add user). Here, note that User=`QTMHHTTP` and Data Authority=`*R`.
5. Repeat steps one through three to authorize QTMHHTTP to the stash file (DEFAULT.sth).

9.5.3 Creating a System Certificate with Your Intranet CA (V4R4)

After creating the intranet CA, DCM prompts you to create a system (or server) certificate. To use Secure Sockets Layer (SSL) for secure Web serving, your server must have a digital certificate. When you create a system certificate in DCM, the system certificate and keys are stored in the following default directory:

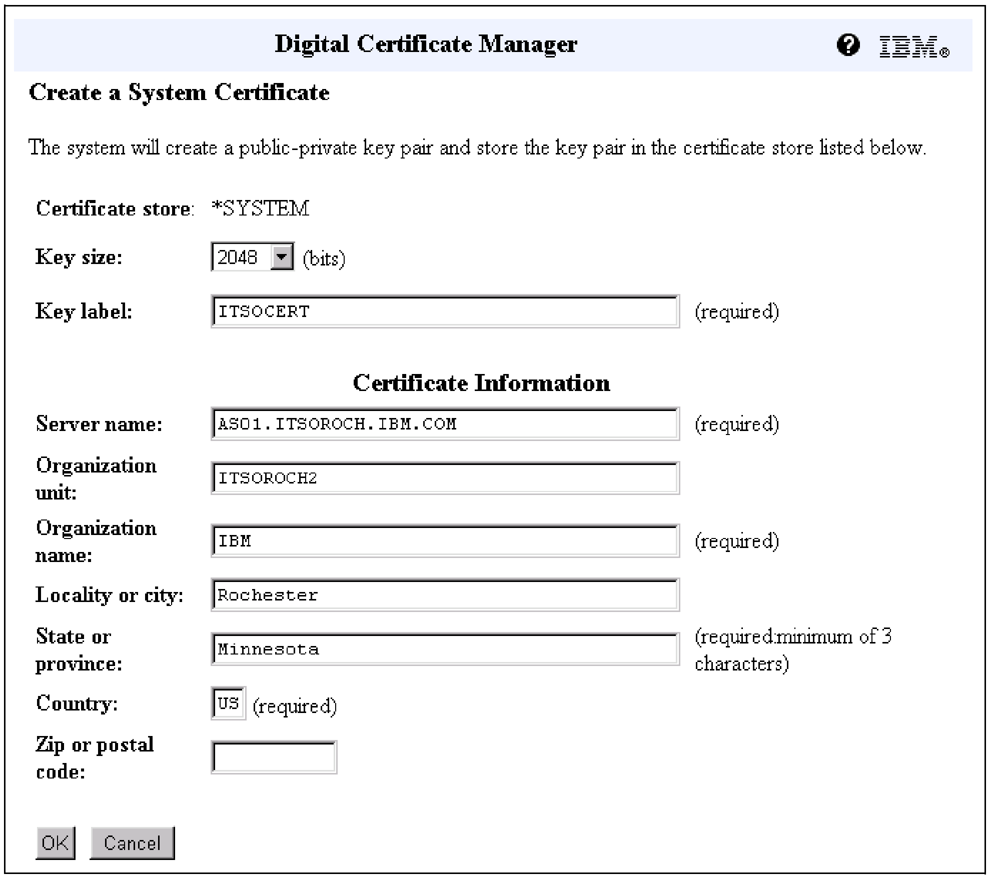
```
/QIBM/USERDATA/ICSS/CERT/SERVER/
```

This is also known as the certificate store `*SYSTEM`.

To create a system certificate with your intranet CA, complete the following steps:

1. Complete the Create a Server Certificate form as shown in Figure 72 on page 130. Replace the field values with your organization's information.

   The options for the key size are determined by the IBM Cryptographic Access Provider (5769-ACx) licensed program product installed on your system. This is the key size that is used to generate your public and private keys.



Digital Certificate Manager

Create a System Certificate

The system will create a public-private key pair and store the key pair in the certificate store listed below.

Certificate store:	*SYSTEM	
Key size:	2048 (bits)	
Key label:	ITSOCERT	(required)

Certificate Information

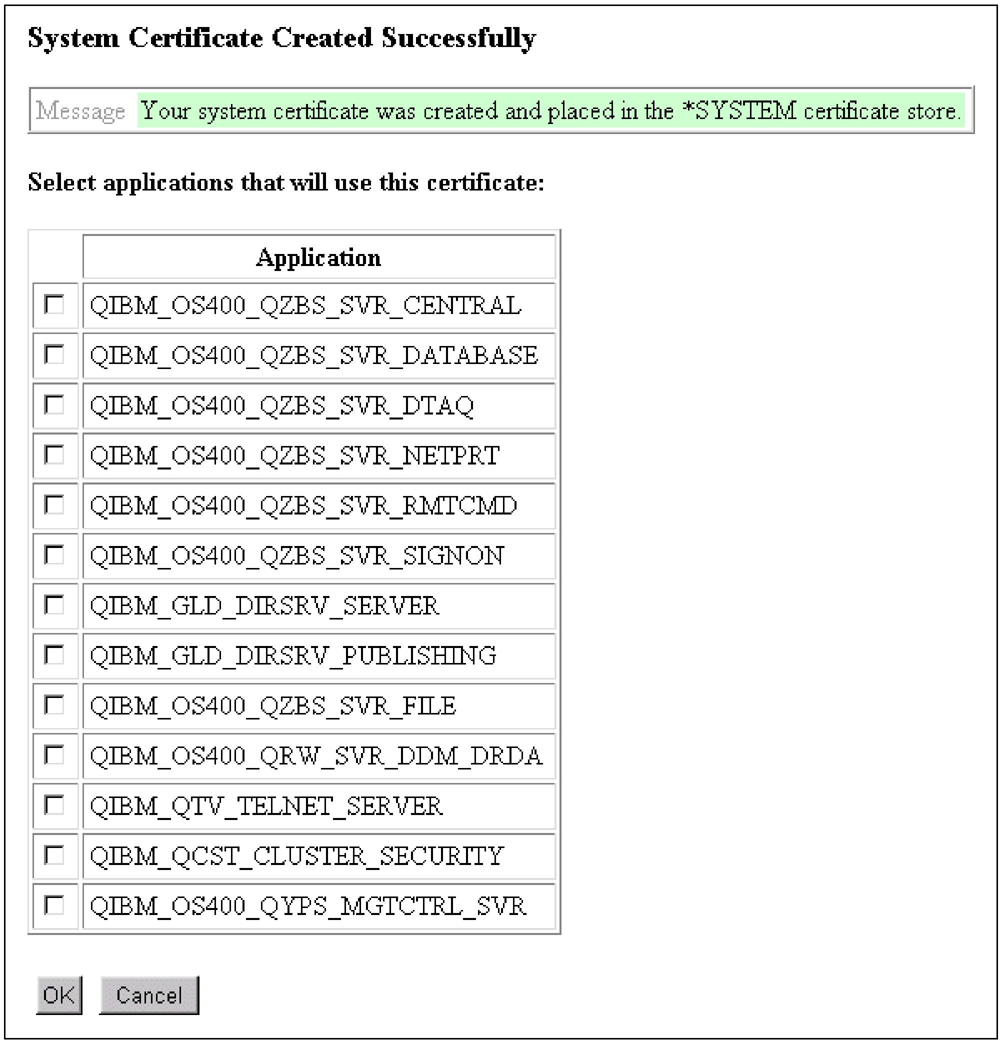
Server name:	AS01.ITSOROCH.IBM.COM	(required)
Organization unit:	ITSOROCH2	
Organization name:	IBM	(required)
Locality or city:	Rochester	
State or province:	Minnesota	(required:minimum of 3 characters)
Country:	US (required)	
Zip or postal code:		

Figure 72. Create a System Certificate Page

By default, the system inserts the fully qualified name of the AS/400 system into the system name field. You can give the server any name. However, the fully qualified TCP/IP host name is usually used for the server name. Some CAs require that the state name be spelled out completely. We recommend always using the entire name rather than a short form.

2. Click **OK**.

The System Certificate Created Successfully page appears (Figure 73 on page 131).


System Certificate Created Successfully

Message Your system certificate was created and placed in the *SYSTEM certificate store.

Select applications that will use this certificate:

	Application
☐	QIBM_OS400_QZBS_SVR_CENTRAL
☐	QIBM_OS400_QZBS_SVR_DATABASE
☐	QIBM_OS400_QZBS_SVR_DTAQ
☐	QIBM_OS400_QZBS_SVR_NETPRT
☐	QIBM_OS400_QZBS_SVR_RMTCMD
☐	QIBM_OS400_QZBS_SVR_SIGNON
☐	QIBM_GLD_DIRSRV_SERVER
☐	QIBM_GLD_DIRSRV_PUBLISHING
☐	QIBM_OS400_QZBS_SVR_FILE
☐	QIBM_OS400_QRW_SVR_DDM_DRDA
☐	QIBM_QTV_TELNET_SERVER
☐	QIBM_QCST_CLUSTER_SECURITY
☐	QIBM_OS400_QYPS_MGTCTRL_SVR

Figure 73. System Certificate Created Successfully Page

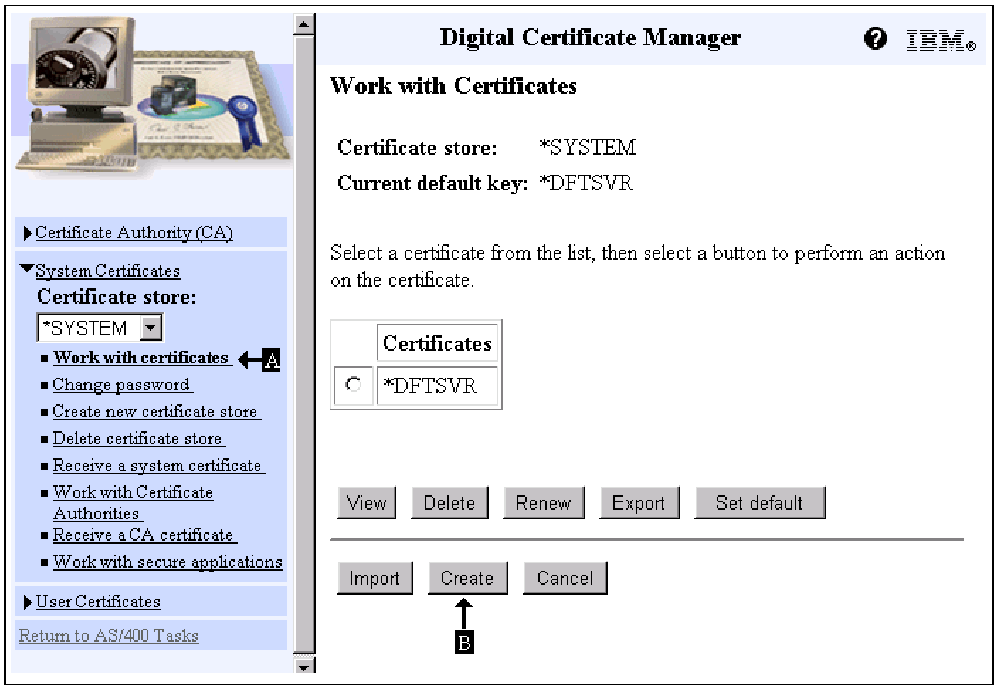
3. From this page (Figure 73), you can select which applications use this system certificate for SSL connections. After you make your selection, click **OK**. A message will appear that confirms that any applications you selected will use this system certificate. Click **Done**.

9.5.3.1 Creating a Server Certificate with an Existing Intranet CA

The steps to create a server certificate described in the previous section assume that you are creating the intranet CA for the first time. If your administrator has already created an intranet CA and server certificate, you can use the existing server certificate in your HTTP server configuration.

Follow these steps to create a new system certificate using an existing intranet CA:

1. Click **System Certificates->Work with certificates** (A) in DCM. The panel shown in Figure 74 on page 132 appears. Enter the certificate store password when prompted.



Digital Certificate Manager

Work with Certificates

Certificate store: *SYSTEM

Current default key: *DFTSVR

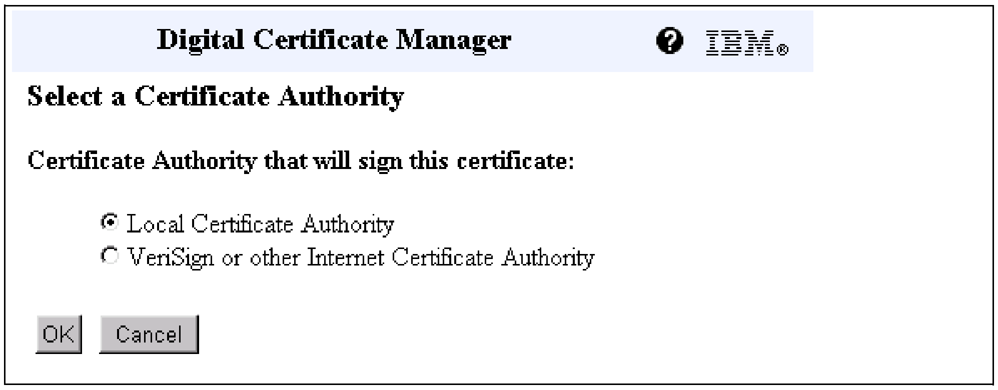
Select a certificate from the list, then select a button to perform an action on the certificate.

	Certificates
○	*DFTSVR

Certificate store: *SYSTEM

Figure 74. Digital Certificate Manager - Work with Certificates

2. Click **Create** (B) to create a new system certificate. The display shown in Figure 75 appears.



Digital Certificate Manager

Select a Certificate Authority

Certificate Authority that will sign this certificate:

[✓ Local Certificate Authority]
VeriSign or other Internet Certificate Authority

Figure 75. Creating a System Certificate with an Existing Intranet CA

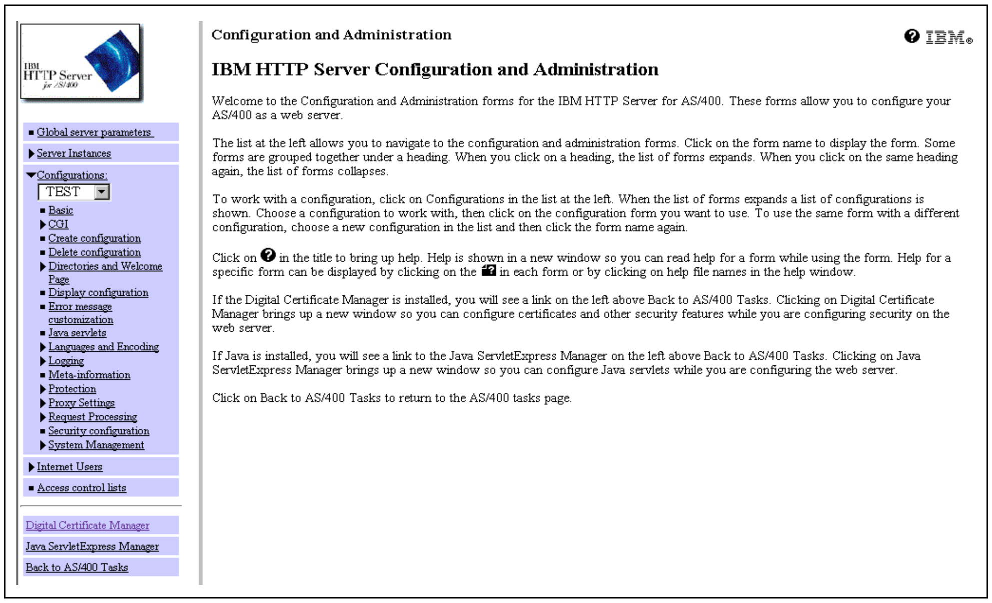
3. Select **Local Certificate Authority**, and click **OK**.

You are now at the same place in the process that the system takes you to when you create a system certificate during the Create CA process. Go to step 1 in 9.5.3, “Creating a System Certificate with Your Intranet CA (V4R4)” on page 129, and complete the procedure found there.

9.5.4 Configuring Web Server to Use SSL Server Authentication (V4R3)

The Web server must be configured to run over SSL and use the server certificate you created in 9.5.2, “Creating a Server Certificate with Your Intranet CA (V4R3)” on page 126. To configure your HTTP server to run over SSL and use a server certificate, perform the following tasks:

1. From Digital Certificate Manager, click **Return to AS/400 Tasks**. The AS/400 Tasks page is displayed as shown in Figure 64 on page 123.
2. Click **IBM HTTP Server for AS/400**.
3. Click **Configuration and Administration**.
4. In the left frame, click **Configurations**.
5. Select your HTTP configuration file in the drop-down box immediately beneath the Configurations link as shown in Figure 76.



Configuration and Administration

IBM HTTP Server Configuration and Administration

Welcome to the Configuration and Administration forms for the IBM HTTP Server for AS/400. These forms allow you to configure your AS/400 as a web server.

The list at the left allows you to navigate to the configuration and administration forms. Click on the form name to display the form. Some forms are grouped together under a heading. When you click on a heading, the list of forms expands. When you click on the same heading again, the list of forms collapses.

To work with a configuration, click on Configurations in the list at the left. When the list of forms expands a list of configurations is shown. Choose a configuration to work with, then click on the configuration form you want to use. To use the same form with a different configuration, choose a new configuration in the list and then click the form name again.

Click on ❓ in the title to bring up help. Help is shown in a new window so you can read help for a form while using the form. Help for a specific form can be displayed by clicking on the ❓ in each form or by clicking on help file names in the help window.

If the Digital Certificate Manager is installed, you will see a link on the left above Back to AS/400 Tasks. Clicking on Digital Certificate Manager brings up a new window so you can configure certificates and other security features while you are configuring security on the web server.

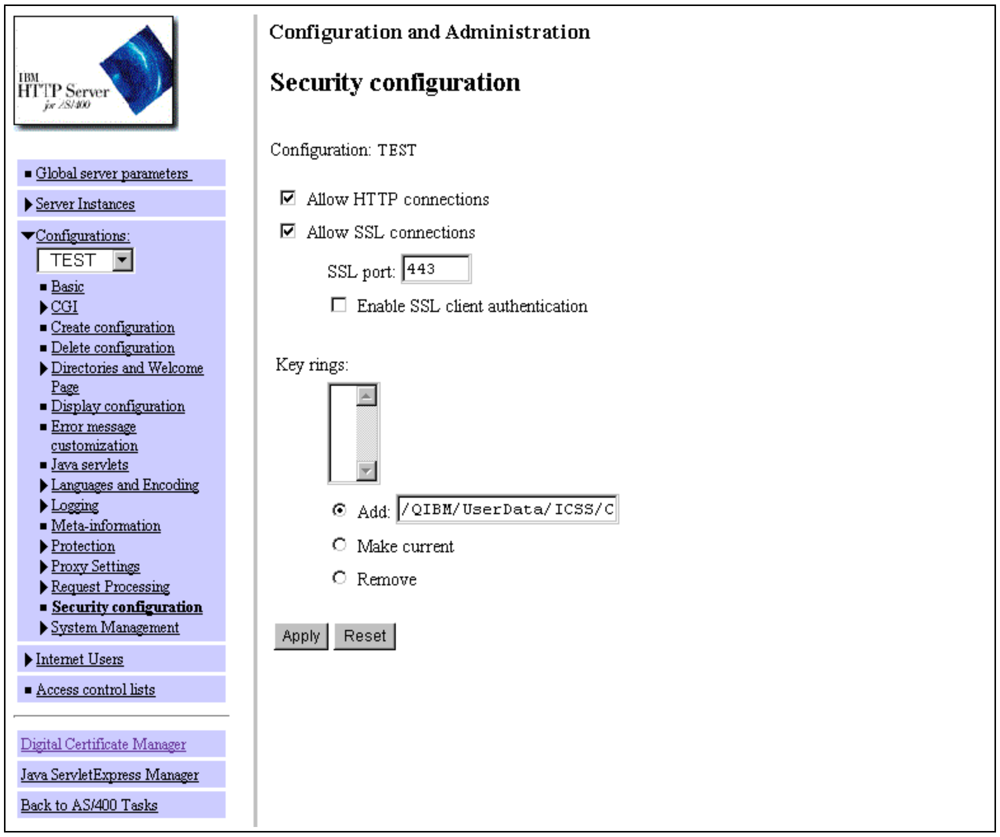
If Java is installed, you will see a link to the Java ServletExpress Manager on the left above Back to AS/400 Tasks. Clicking on Java ServletExpress Manager brings up a new window so you can configure Java servlets while you are configuring the web server.

Click on Back to AS/400 Tasks to return to the AS/400 tasks page.

Figure 76. HTTP Server Configuration

6. Click on **Security configuration**. Complete the Security configuration page (Figure 77 on page 134).
   a. Check **Allow SSL connections**.
   b. Accept the default SSL port (`443`), or specify the port you wish to use for SSL.
   c. De-select **Enable SSL client authentication**.
   d. Add the key ring path and file name. If you copied it to the clipboard, you can paste it now. If not, the key ring and file name is:

   ```
   /QIBM/USERDATA/ICSS/CERT/SERVER/DEFAULT.KYR
   ```



Configuration and Administration

Security configuration

Configuration: TEST

[✓] Allow HTTP connections

[✓] Allow SSL connections

SSL port: 443

[ ] Enable SSL client authentication

Key rings:

(●) Add: /QIBM/UserData/ICSS/C

( ) Make current

( ) Remove

Figure 77. Security Configuration Page

e. Click **Apply**.

You should see this message at the top of the screen:

```
The configuration file was successfully updated. Server instances that
are using this configuration must be stopped and started for the changes
to take affect.
```

You should also see your key ring file added in the Key rings box.

7. Stop the server instance and start it again. In the left window pane, click **Server Instances**.
8. Click **Work with server instances**.
9. From the drop-down box, select your server instance (Figure 78 on page 135).

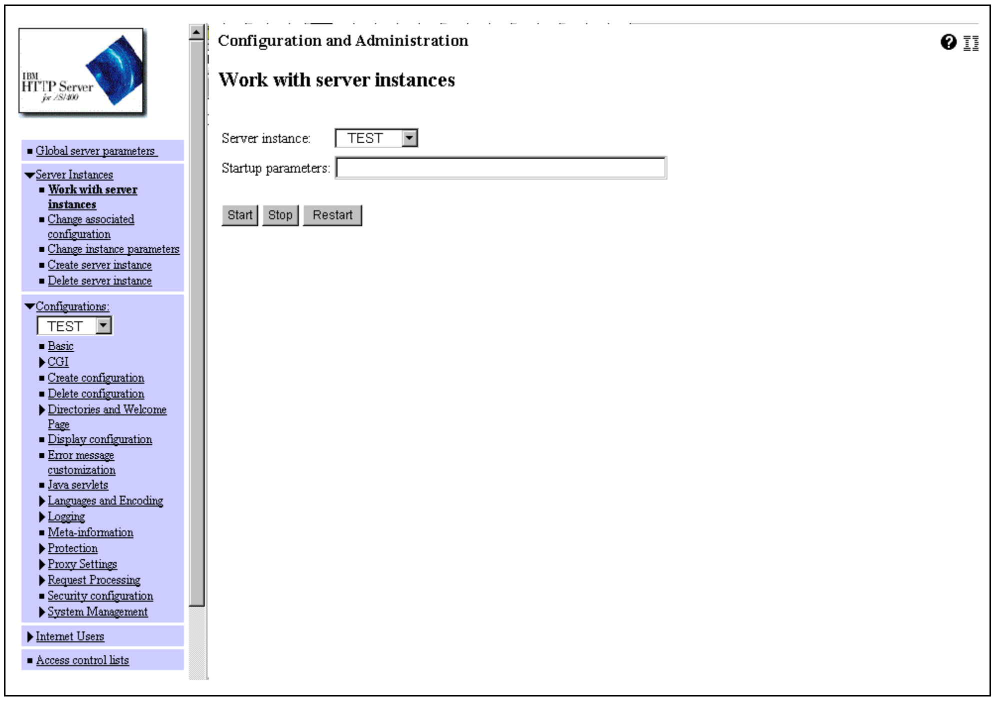


Figure 78. Work with Server Instances

Click **Stop**. Wait until you see this message at the top of your window:

```
The server instance was successfully stopped.
```

10.From the drop-down box, select your server instance (Figure 78).

Click **Start**.

You should see this message:

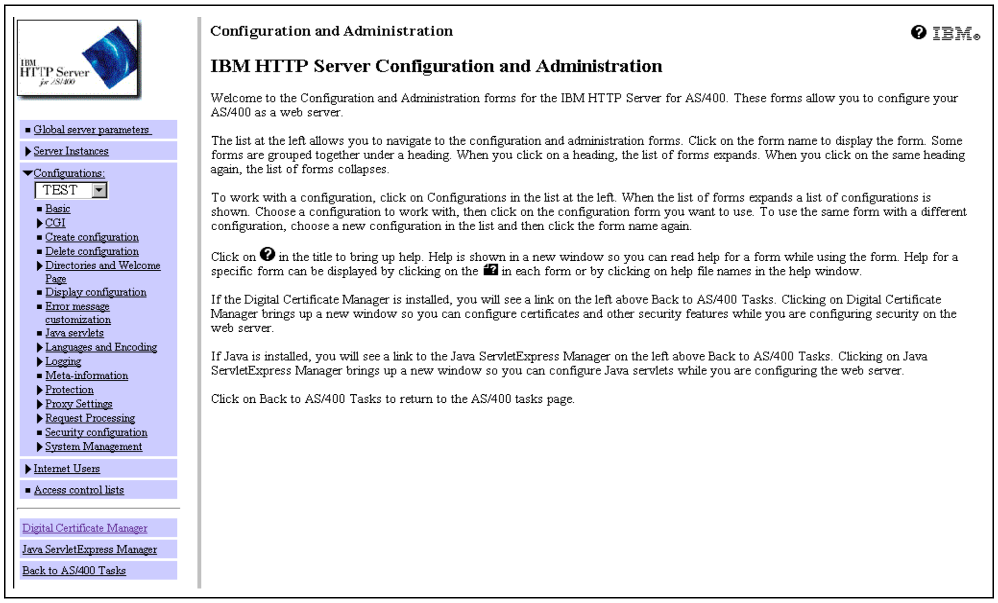
```
The server instance was successfully started.
```

You have now successfully configured your Web server to use SSL with server authentication.

9.5.5 Configuring Web Server to Use SSL Server Authentication (V4R4)

The Web server must be configured to run over SSL and use the server certificate you created in 9.5.3, “Creating a System Certificate with Your Intranet CA (V4R4)” on page 129. To configure your HTTP server to run over SSL and use a server certificate, you must perform the following tasks:

1. From Digital Certificate Manager, click **Return to AS/400 Tasks**. The AS/400 Tasks page is displayed (Figure 64 on page 123).
2. Click **IBM HTTP Server for AS/400**.
3. Click **Configuration and Administration**.
4. In the left frame, click **Configurations**.
5. Select your HTTP configuration file in the drop-down box located beneath the Configurations link as shown in Figure 79 on page 136.



IBM HTTP Server Configuration and Administration

Welcome to the Configuration and Administration forms for the IBM HTTP Server for AS/400. These forms allow you to configure your AS/400 as a web server.

The list at the left allows you to navigate to the configuration and administration forms. Click on the form name to display the form. Some forms are grouped together under a heading. When you click on a heading, the list of forms expands. When you click on the same heading again, the list of forms collapses.

To work with a configuration, click on Configurations in the list at the left. When the list of forms expands a list of configurations is shown. Choose a configuration to work with, then click on the configuration form you want to use. To use the same form with a different configuration, choose a new configuration in the list and then click the form name again.

Click on ❓ in the title to bring up help. Help is shown in a new window so you can read help for a form while using the form. Help for a specific form can be displayed by clicking on the ❓ in each form or by clicking on help file names in the help window.

If the Digital Certificate Manager is installed, you will see a link on the left above Back to AS/400 Tasks. Clicking on Digital Certificate Manager brings up a new window so you can configure certificates and other security features while you are configuring security on the web server.

If Java is installed, you will see a link to the Java ServletExpress Manager on the left above Back to AS/400 Tasks. Clicking on Java ServletExpress Manager brings up a new window so you can configure Java servlets while you are configuring the web server.

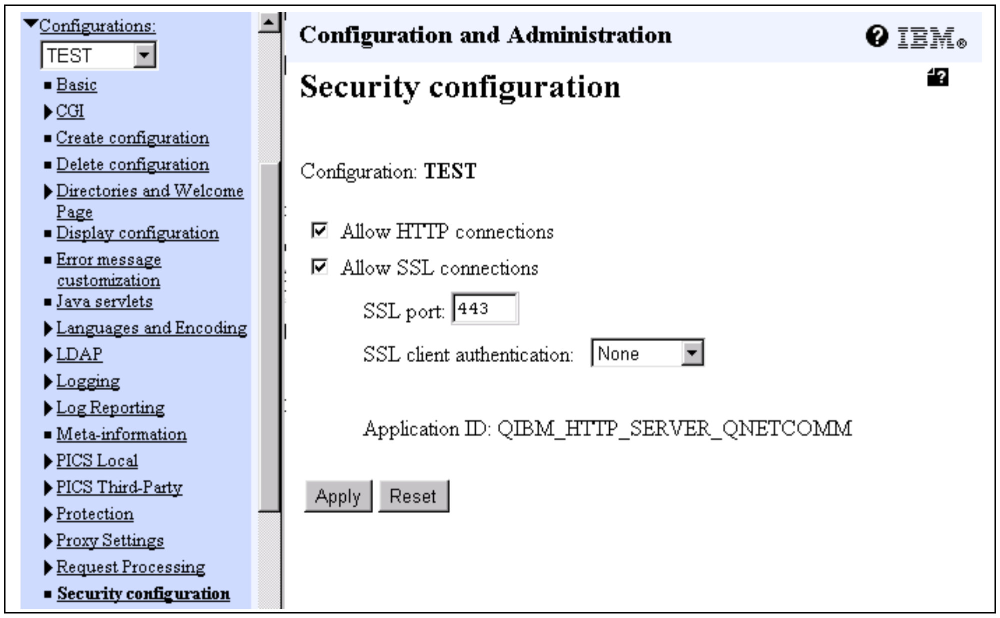
Click on Back to AS/400 Tasks to return to the AS/400 tasks page.

Figure 79. HTTP Server Configuration

6. Click on **Security configuration**. The display shown in Figure 80 on page 137 appears.
7. Complete the Security configuration page.
   a. Check **Allow SSL connections**.
   b. Accept the default SSL port (`443`), or specify the port you wish to use for SSL.

Note

The **Application ID** value on the page (Figure 80 on page 137) may show one of several values. The value may be `No ID created yet` or `QIBM_HTTP_SERVER_xxxxxxxx`, where `xxxxxxxx` is the name of this configuration or the name of the configuration from which this configuration was built. Even if all the values are specified correctly, you should still click **Apply** in the next step.



Security configuration

Configuration: TEST

☑ Allow HTTP connections

☑ Allow SSL connections

SSL port: 443

SSL client authentication: None

Application ID: QIBM_HTTP_SERVER_QNETCOMM

Figure 80. Security Configuration Page

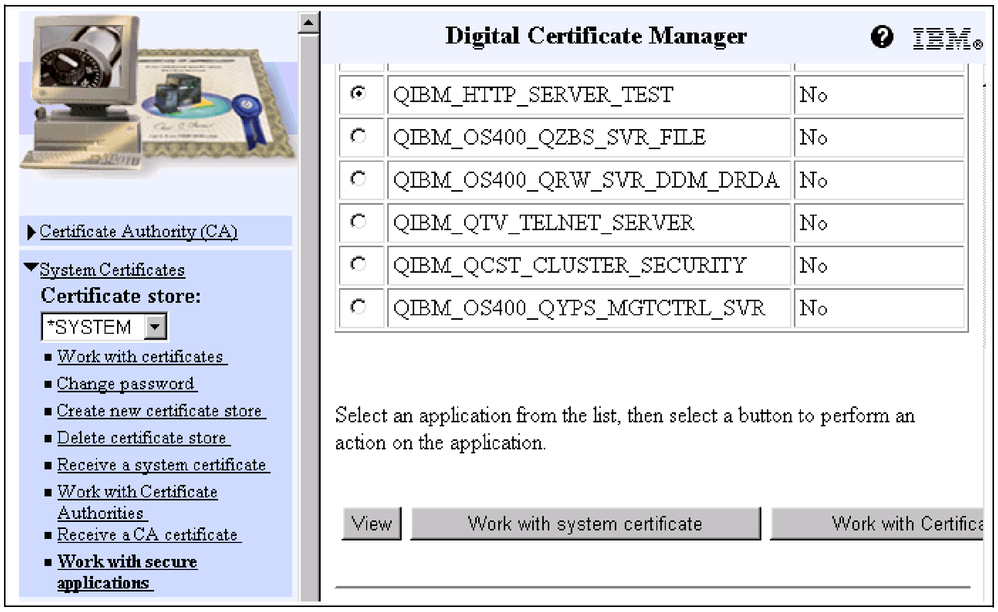
c. Click **Apply**.

You should see this message at the top of the screen:

```
The configuration file was successfully updated. Server instances that are
using this configuration must be stopped and started for the changes to take
affect.
```

You should also see the **Application ID** added or changed to the value `QIBM_HTTP_SERVER_xxxxxxxx`, where `xxxxxxxx` is the name of this configuration (TEST in this example). Record the **Application ID**. You will need to know it to complete the SSL configuration. You now need to enable this HTTP server configuration as a secure application in Digital Certificate Manager (DCM).

8. In the left frame, click **Digital Certificate Manager.** The DCM welcome page is shown in a new browser window.
9. Click **System Certificates->Work with secure applications** in DCM. The panel shown in Figure 81 on page 138 appears. Enter the certificate store password when prompted.

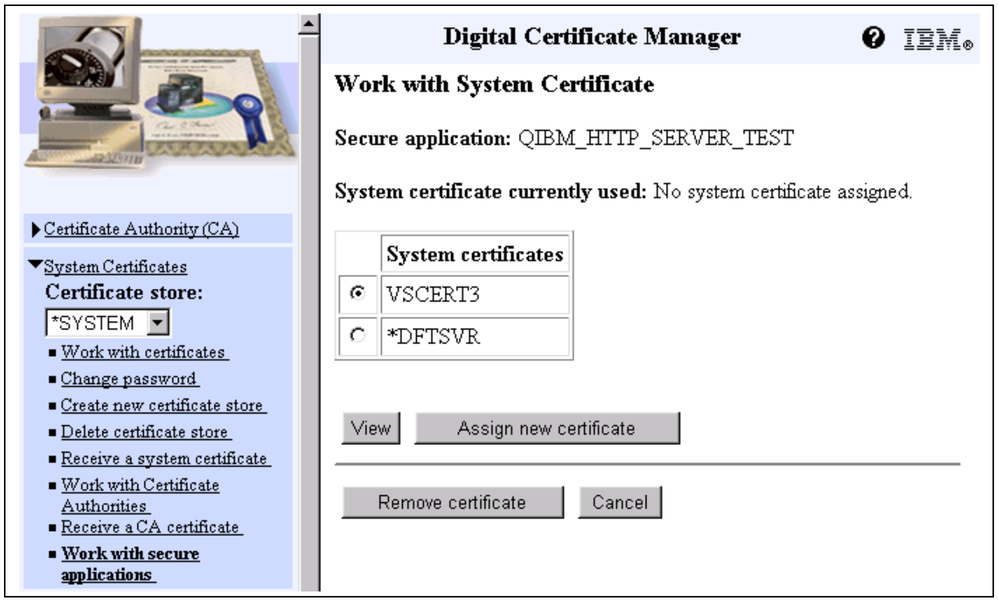

Digital Certificate Manager
IBM
Certificate Authority (CA)
System Certificates
Certificate store:
*SYSTEM
Work with certificates
Change password
Create new certificate store
Delete certificate store
Receive a system certificate
Work with Certificate Authorities
Receive a CA certificate
Work with secure applications
View
Work with system certificate
Work with Certifica

●	QIBM_HTTP_SERVER_TEST	No
○	QIBM_OS400_QZBS_SVR_FILE	No
○	QIBM_OS400_QRW_SVR_DDM_DRDA	No
○	QIBM_QTV_TELNET_SERVER	No
○	QIBM_QCST_CLUSTER_SECURITY	No
○	QIBM_OS400_QYPS_MGTCTRL_SVR	No

Select an application from the list, then select a button to perform an action on the application.

Figure 81. Work with Secure Applications in DCM

10. Scroll down the list of applications until you find the Application ID you want to secure. Select the application, and click **Work with system certificate** in the right frame. This allows you to select which certificate will be used for SSL. The display shown in Figure 82 appears.



Work with System Certificate

Secure application: QIBM_HTTP_SERVER_TEST

System certificate currently used: No system certificate assigned.

	System certificates
●	VSCERT3
○	*DFTSVR

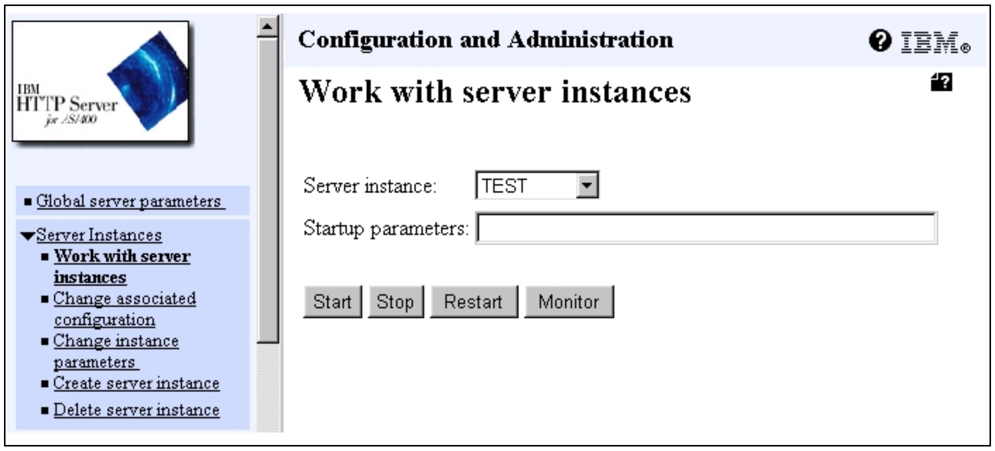
Figure 82. Work with System Certificate

11. An application may only use one certificate. However, one certificate may be used with multiple applications. You may use the **View** button to display the details about the certificates available. Select a system certificate from the list to be used with this application (VSCERT3 in this example), and click **Assign new certificate**. A message appears, which states:

```
The system certificate was assigned to the application.
```

When the certificate is assigned the CA that issued the certificate, it is set as a trusted root for the application. You can use the **Work with Certificate Authority** button shown in Figure 81 on page 138 to check the CA assignment for an application.

12. Click **OK**. The display shown in Figure 81 on page 138 appears. You should now stop the server instance, and start it again.
13. Return to the browser window that contains the **IBM HTTP Server Configuration and Administration** page (Figure 79 on page 136). In the left window pane, click **Server Instances**.
14. Click **Work with server instances**.
15. From the drop-down box, select your server instance (Figure 83).


IBM HTTP Server for AS/400
Global server parameters
Server Instances
Work with server instances
Change associated configuration
Change instance parameters
Create server instance
Delete server instance
Configuration and Administration
IBM
Work with server instances
Start
Stop
Restart
Monitor

Server instance: TEST
Startup parameters:

Figure 83. Work with Server Instances

Click **Stop**. Wait until you see this message at the top of your window:

```
The server instance was successfully stopped.
```

16. From the drop-down box, select your server instance (see Figure 83).

    Click **Start**.

    You should see this message:

```
The server instance was successfully started.
```

You have now successfully configured your Web server to use SSL with server authentication.

9.6 Requesting a Server Certificate from an Internet CA

This section describes how to obtain a server certificate from an Internet certificate authority.

To conduct commercial business on the Internet, you should request your server certificate from an Internet certificate authority. For example, you may consider a CA, such as VeriSign or Thawte, which are widely known by clients, browsers, and servers.

For your private Web network within your own company, university, or group, or for testing purposes, using Digital Certificate Manager (DCM) lets you act as your own CA. Section 9.5, “Creating a Self-signed Certificate” on page 122, explains this procedure.

To use a server certificate issued by an Internet CA, perform these steps:

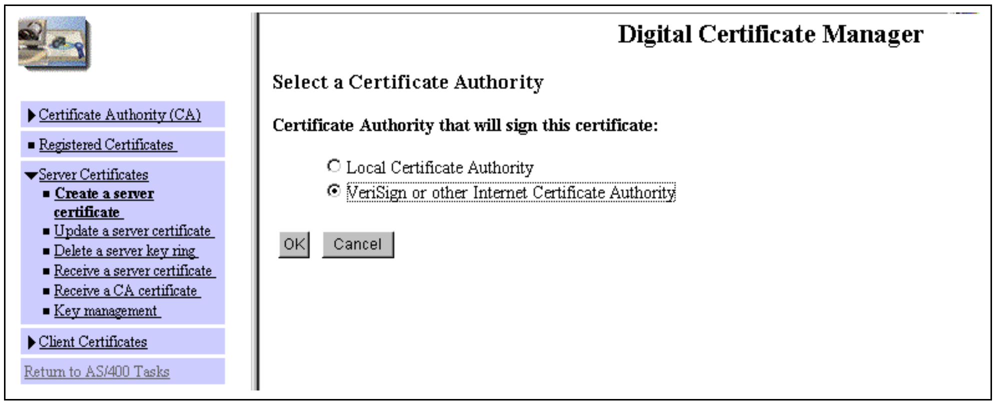
1. Request the server certificate from an Internet CA.
2. Receive a server certificate for this server.
3. Configure the HTTP server to use SSL and server authentication.

9.6.1 Requesting a Server Certificate from an Internet CA (V4R3)

To use SSL for secure Web serving, your server must have a digital certificate. You can use an intranet CA to issue a server certificate, or you can use an Internet CA. Refer to 9.5, “Creating a Self-signed Certificate” on page 122, for more information.

When you choose to use an Internet CA to issue a server certificate, you must first request the certificate. Follow these steps:

1. From the Digital Certificate Manager (DCM) page, click **Server Certificates** in the left-hand frame to display an extended list of server tasks.
2. Click on **Create a server certificate** from the list to display the Select a Certificate Authority page.
3. Select **VeriSign or other Internet Certificate Authority** as shown in Figure 84.



Digital Certificate Manager

Select a Certificate Authority

Certificate Authority that will sign this certificate:

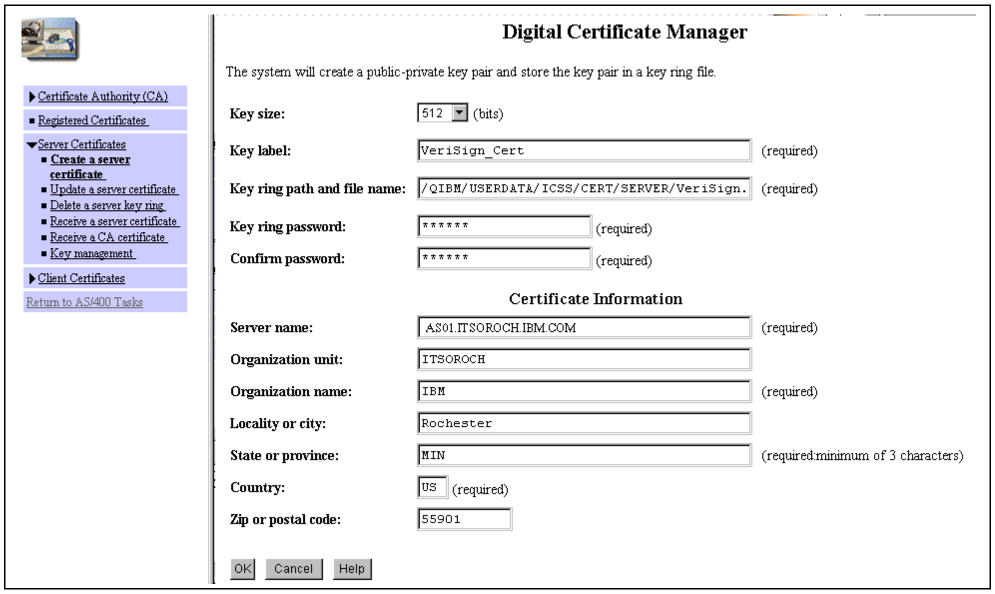
Local Certificate Authority [✓ VeriSign or other Internet Certificate Authority]

Figure 84. Requesting a Certificate from VeriSign or Other Internet CA

Click **OK** to display the Create a Server Certificate form.

4. Complete the Create a Server Certificate form as shown in Figure 85 on page 141. Replace the field values with your organization’s information.

   The options for the key size are determined by the IBM Cryptographic Access Provider (5769-ACx) licensed program product installed on your system. This is the key size that will be used to generate your public and private keys.



Digital Certificate Manager

The system will create a public-private key pair and store the key pair in a key ring file.

Key size:	512 (bits)	
Key label:	VeriSign_Cert	(required)
Key ring path and file name:	/QIBM/USERDATA/ICSS/CERT/SERVER/VeriSign.	(required)
Key ring password:	******	(required)
Confirm password:	******	(required)

Certificate Information

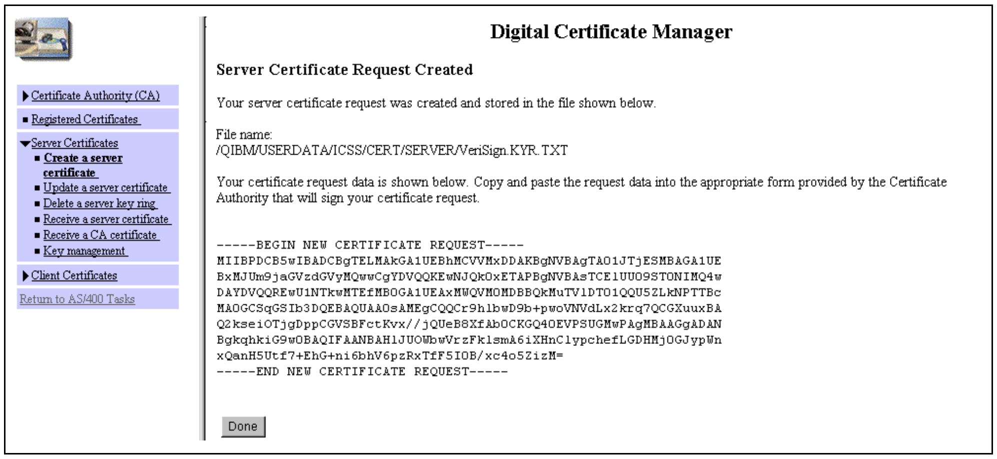
Server name:	AS01.ITSOROCH.IBM.COM	(required)
Organization unit:	ITSOROCH	
Organization name:	IBM	(required)
Locality or city:	Rochester	
State or province:	MIN	(required:minimum of 3 characters)
Country:	US	(required)
Zip or postal code:	55901	

Figure 85. Requesting a Server Certificate from an Internet CA

By default, the system inserts the fully qualified name of the AS/400 system into the system name field. *Do not change this name.* This is the name used to describe your server. You can give the server any name, although the fully qualified TCP/IP host name is usually used for the server name.

5. Click **OK** to process the Create a Certificate Request form.

   You receive the Server Certificate Request Created page as shown in Figure 86.



Digital Certificate Manager

Server Certificate Request Created

Your server certificate request was created and stored in the file shown below.

File name:
/QIBM/USERDATA/ICSS/CERT/SERVER/VeriSign.KYR.TXT

Your certificate request data is shown below. Copy and paste the request data into the appropriate form provided by the Certificate Authority that will sign your certificate request.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBPDCB5wIBADCBgTELMAkGA1UEBhMCVVMxDDAKBgNVBAgTA01JTjESMBAGA1UE
BxMJUm9jaGVzdGVyMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsTCE1UU09STONIMQ4w
DAYDVQQREwU1NTkwMTEfMB0GA1UEAxMWQVMOMDBBQkMuTV1DT01QQU5ZLkNPTTBc
MA0GCSqGSIb3DQEBAQUAA0sAMEgCQQCr9hlbwD9b+pwoVNVdLx2krq7QCGXuuxBA
Q2kseiOTjgDppCGVSBFctKvx//jQUeB8XfAbOCKGQ4OEVPSUGMwPAgMBAAGgADAN
BgkqhkiG9w0BAQIFAANBAHlJUOWbwVrzFklsmA6iXHnClypchefLGDHMjOGJypWn
xQanH5Utf7+EhG+ni6bhV6pzRxTfF5IOB/xc4o5ZizM=
-----END NEW CERTIFICATE REQUEST-----
```

Figure 86. Server Certificate Request Generated by DCM

Note

Do not click Done or close the browser yet. You need to cut and paste the certificate request when you submit the Certificate Signing Request to the Internet CA.

6. Copy the Server Certificate Request to your clipboard. Start at `-----BEGIN NEW CERTIFICATE REQUEST-----` and end at `-----END NEW CERTIFICATE REQUEST-----`. Click **Done** to close the page.
7. Follow your Internet CA procedures to paste the certificate request. For example, to request a certificate from VeriSign, follow the instructions that are described at the following Web site: `http://www.verisign.com`

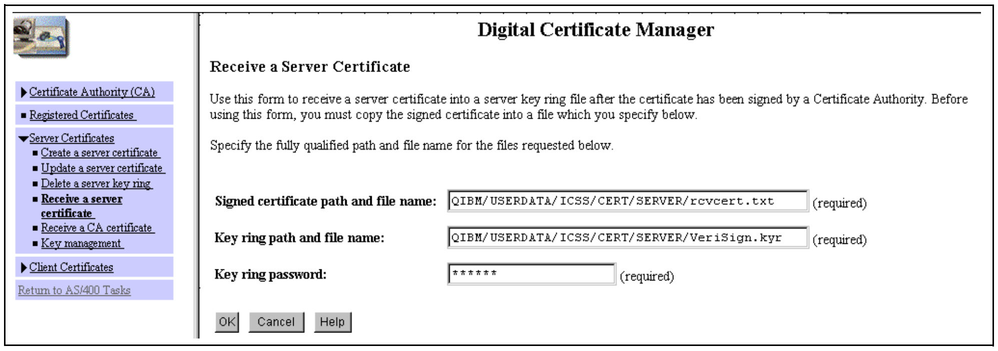
   When VeriSign is satisfied that you have met all of its requirements, it e-mails the secure server certificate to you. You should receive it in three to five business days. Other certificate authorities have their own procedures.

9.6.2 Receiving a Server Certificate for This Server (V4R3)

After you receive the certificate from the Internet CA, copy the signed server certificate to a text file that DCM can access when you perform the Receive server certificate task. Perform the following steps:

1. Copy the signed server certificate presented to you by the Internet CA to your clipboard. Start at `-----BEGIN CERTIFICATE REQUEST-----`, and end at `-----END CERTIFICATE REQUEST-----`.
2. Paste the signed server certificate in your clipboard into a .txt file. Use a text editor of your choice, for example, Notepad, to create a .txt file and paste the server certificate issued by the Internet CA.
3. Save the file in your AS/400 system IFS. Use a mapped network drive and save the .txt file that contains the server certificate issued by the Internet CA in the following path (enter a file name of your choice):

   `/QIBM/USERDATA/ICSS/CERT/SERVER/rcvcert.txt`
4. In DCM, click **Receive a server certificate**, and complete the Receive a Server Certificate page (Figure 87).



Receive a Server Certificate

Use this form to receive a server certificate into a server key ring file after the certificate has been signed by a Certificate Authority. Before using this form, you must copy the signed certificate into a file which you specify below.

Specify the fully qualified path and file name for the files requested below.

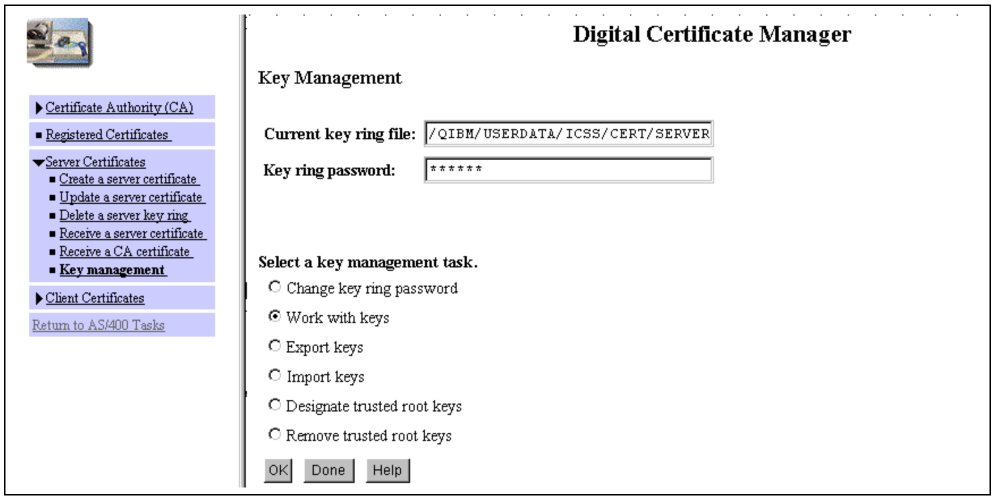
Signed certificate path and file name: QIBM/USERDATA/ICSS/CERT/SERVER/rcvcert.txt (required)

Key ring path and file name: QIBM/USERDATA/ICSS/CERT/SERVER/VeriSign.kyr (required)

Key ring password: ****** (required)

Figure 87. Receiving a Server Certificate Issued by an Internet CA

5. The Certificate Received page is displayed. You have the option to use the received certificate with the ADMIN or LDAP server. *Do not select these options*. Click **OK**.
6. You should receive a Server Configuration Status message indicating the server certificate operations are complete. Click **Done**.
7. You must now set the key as the default key. In DCM, click **Key management**. Complete the Key Management page, and select **Work with keys** (Figure 88 on page 143).



Digital Certificate Manager

Key Management

Current key ring file: /QIBM/USERDATA/ICSS/CERT/SERVER

Key ring password: ******

Select a key management task.

- ( ) Change key ring password
- (•) Work with keys
- ( ) Export keys
- ( ) Import keys
- ( ) Designate trusted root keys
- ( ) Remove trusted root keys

Figure 88. Key Management Page

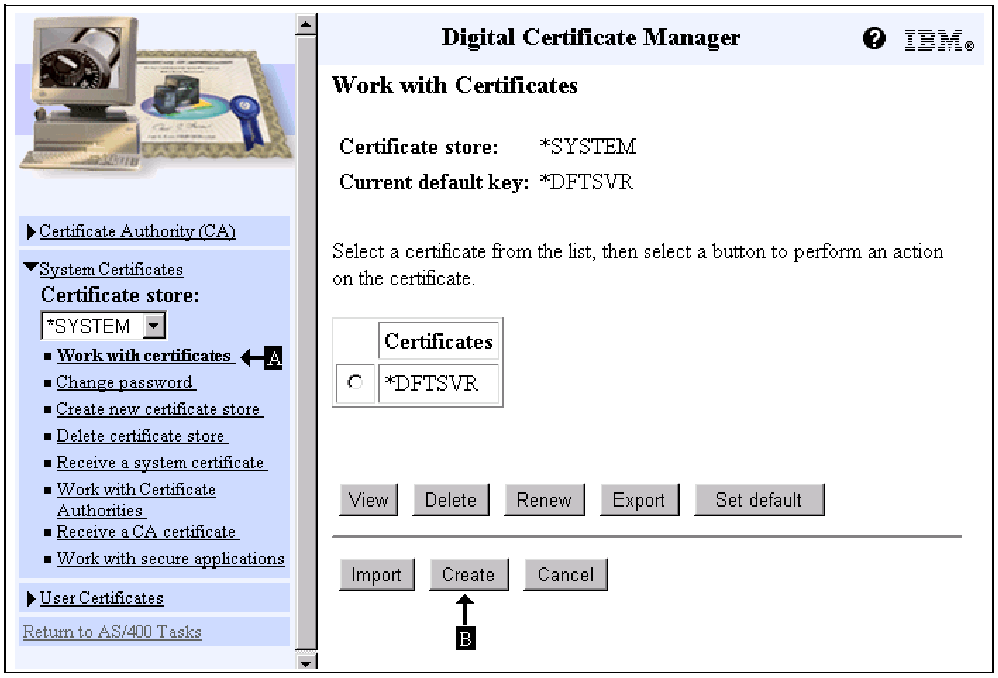
8. Select the key with the label corresponding to the certificate you received from the Internet CA (VeriSign_Cert in our example). Select **Set key** to be the default, and click **OK**.

9.6.3 Requesting a System Certificate from an Internet CA (V4R4)

To use SSL for secure Web serving, your system must have a digital certificate. You can use an intranet CA to issue a system certificate, or you can use an Internet CA. Refer to 9.5, “Creating a Self-signed Certificate” on page 122, for more information.

When you choose to use an Internet CA to issue a system certificate, you must first request the certificate. Follow these steps:

1. Click **System Certificates->Work with certificates** (A) in DCM. The panel shown in Figure 89 on page 144 appears. Enter the certificate store password when prompted.



Digital Certificate Manager

Work with Certificates

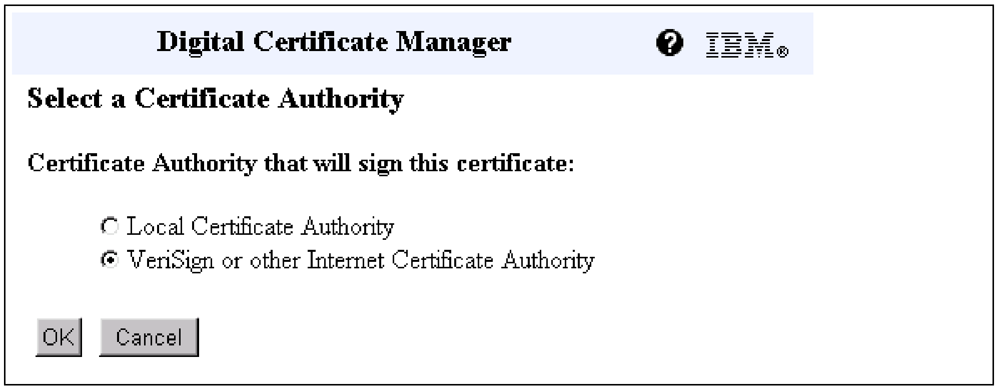
Certificate store: *SYSTEM
Current default key: *DFTSVR

Select a certificate from the list, then select a button to perform an action on the certificate.

	Certificates
○	*DFTSVR

Figure 89. Requesting a Certificate from VeriSign or Other Internet CA

2. Click **Create** (B) to create a new system certificate. The display shown in Figure 90 appears.



Digital Certificate Manager

Select a Certificate Authority

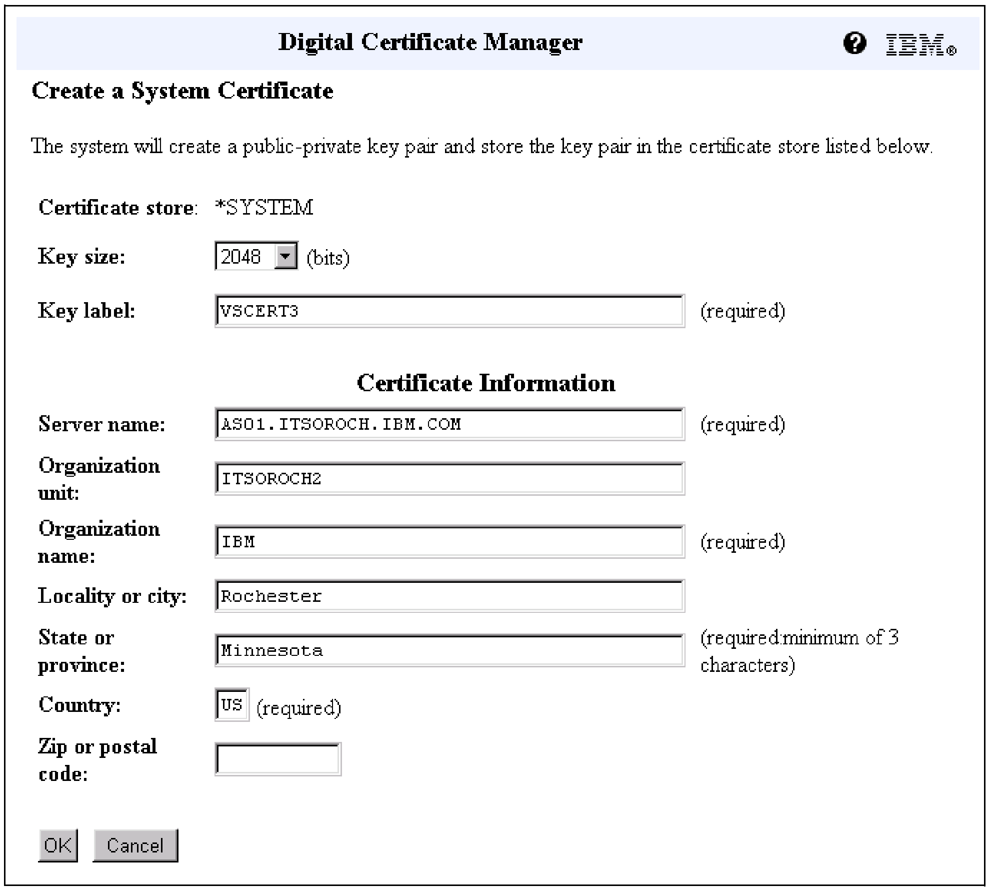
Certificate Authority that will sign this certificate:

○ Local Certificate Authority
● VeriSign or other Internet Certificate Authority

Figure 90. Creating a System Certificate with an Internet CA

3. Select **VeriSign or other Internet Certificate Authority**, and click **OK**.
4. Complete the Create a Server Certificate form as shown in Figure 72 on page 130. Replace the field values with your organization's information.

   The options for the key size are determined by the IBM Cryptographic Access Provider (5769-ACx) licensed program product installed on your system. This is the key size that is used to generate your public and private keys.


Digital Certificate Manager

Create a System Certificate

The system will create a public-private key pair and store the key pair in the certificate store listed below.

Certificate store:	*SYSTEM	
Key size:	2048 (bits)	
Key label:	VSCERT3	(required)

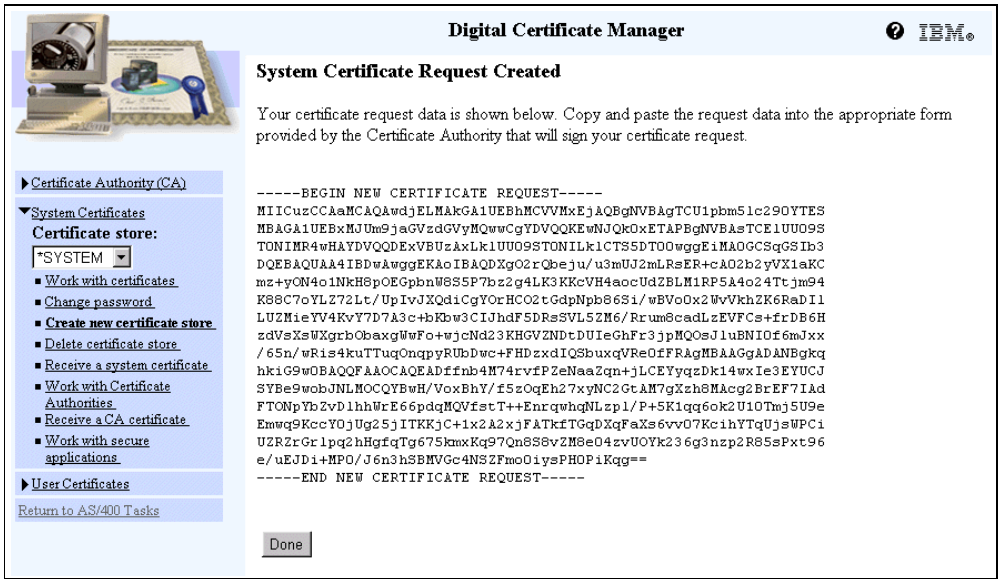
Certificate Information

Server name:	AS01.ITSOROCH.IBM.COM	(required)
Organization unit:	ITSOROCH2	
Organization name:	IBM	(required)
Locality or city:	Rochester	
State or province:	Minnesota	(required:minimum of 3 characters)
Country:	US (required)	
Zip or postal code:		

Figure 91. Create a System Certificate Page

By default, the system inserts the fully qualified name of the AS/400 system into the system name field. *Do not change this name.* This is the name used to describe your server. You can give the server any name. However, the fully qualified TCP/IP host name is usually used for the server name.

5. Click **OK**. The System Certificate Request Created page appears (Figure 92 on page 146).



System Certificate Request Created

Your certificate request data is shown below. Copy and paste the request data into the appropriate form provided by the Certificate Authority that will sign your certificate request.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICuzCCAaMCAQAwdjELMAkGA1UEBhMCVVMxEjAQBgNVBAgTCU1pbm5lc29OYTES
MBAGA1UEBxMJUm9jaGVzdGVyMQwwCgYDVQQKEwNJQk0xETAPBgNVBAsTCE1UUO9S
TONIMR4wHAYDVQQDExVBUzAxLklUUO9STONILklCTS5DTOOwggEiMA0GCSqGSIb3
DQEBAQUAA4IBDwAwggEKAoIBAQDXgO2rQbeju/u3mUJ2mLRsER+cAO2b2yVX1aKC
mz+yON4o1NkH8pOEGpbnW8S5P7bz2g4LK3KKcVH4aocUdZBLM1RP5A4o24Ttjm94
K88C7oYLZ72Lt/UpIvJXQdiCgYOrHCO2tGdpNpb86Si/wBVoOx2WvVkhZK6RaDIl
LUZMieYV4KvY7D7A3c+bKbw3CIJhdF5DRsSVL5ZM6/Rrum8cadLzEVFCs+frDB6H
zdVsXsWXgrbObaxgWwFo+wjcNd23KHGVZNDtDUIeGhFr3jpMQOsJluBNIOf6mJxx
/65n/wRis4kuTTuqOnqpyRUbDwc+FHDzxdIQSbuxqVReOfFRAgMBAAGgADANBgkq
hkiG9w0BAQQFAAOCAQEADffnb4M74rvfPZeNaaZqn+jLCEYyqzDk14wxIe3EYUCJ
SYBe9wobJNLMOCQYBwH/VoxBhY/f5zOqEh27xyNC2GtAM7gXzh8MAcg2BrEF7IAd
FTONpYbZvDlhhWrE66pdqMQVfstT++EnrqwhqNLzpl/P+5K1qq6ok2U1OTmj5U9e
Emwq9KccYOjUg25jITKKjC+1x2A2xjFATkfTGqDXqFaXs6vv07KcihYTqUjsWPCi
UZRZrGrlpq2hHgfqTg675kmxKq97Qn8S8vZM8e04zvUOYk236g3nzp2R85sPxt96
e/uEJDi+MPO/J6n3hSBMVGc4NSZFmoOiysPHOPiKqg==
-----END NEW CERTIFICATE REQUEST-----
```

Figure 92. System Certificate Request Created Page

Note

Do not click Done or close the browser yet. You need to cut and paste the certificate request when you submit the Certificate Signing Request to the Internet CA.

6. Copy the Server Certificate Request to your clipboard. Start at `-----BEGIN NEW CERTIFICATE REQUEST-----` and end at `-----END NEW CERTIFICATE REQUEST-----`. Click **Done** to close the page.
7. Follow your Internet CA procedures to paste the certificate request. For example, to request a certificate from VeriSign, follow the instructions that are described at the Web site: `http://www.verisign.com`

   When VeriSign is satisfied that you have met all of its requirements, it e-mails the secure server certificate to you. You should receive it in three to five business days. Other certificate authorities have their own procedures.

9.6.4 Receiving a System Certificate (V4R4)

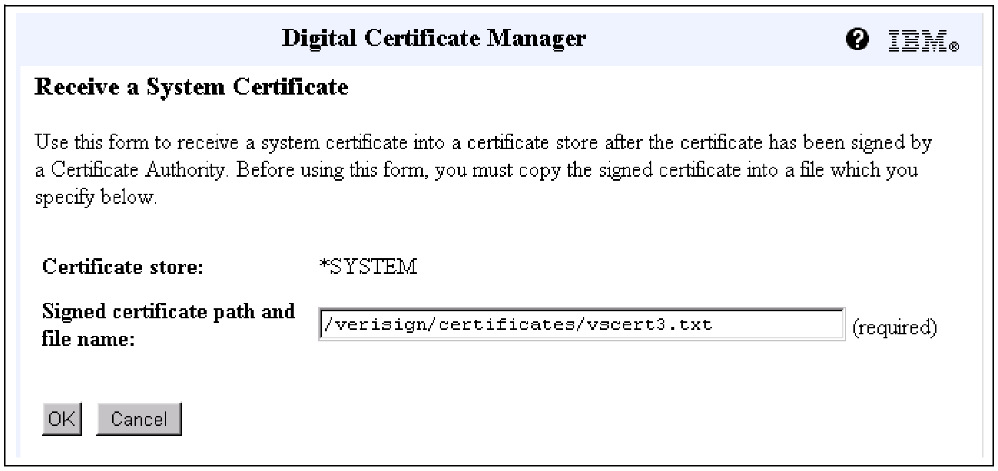
After you receive the certificate from the Internet CA, you need to copy the signed server certificate to a text file that DCM can access when you perform the Receive server certificate task. Perform the following steps:

1. Copy the signed server certificate presented to you by the Internet CA to your clipboard. Start at `-----BEGIN CERTIFICATE REQUEST-----`, and end at `-----END CERTIFICATE REQUEST-----`.
2. Paste the signed system certificate from your clipboard into a .txt file. Use a text editor of your choice, for example, Notepad, to create a .txt file and paste the server certificate issued by the Internet CA.
3. Save the file in your AS/400 system IFS. Use a mapped network drive, and save the .txt file that contains the server certificate issued by the Internet CA.

In our example, we created a directory structure and file with the following path:

```
/verisign/certificates/vscert3.txt
```

4. In DCM, click **Receive a system certificate**. The display shown in Figure 93 appears.



Digital Certificate Manager

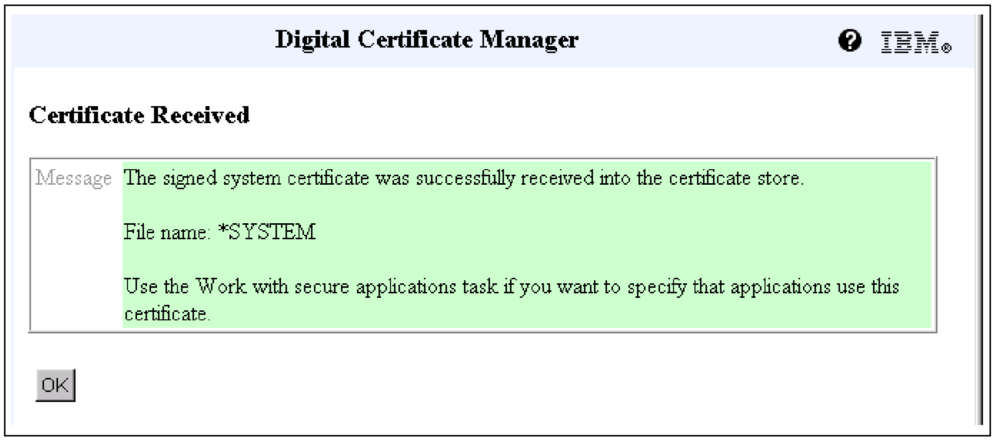
Receive a System Certificate

Use this form to receive a system certificate into a certificate store after the certificate has been signed by a Certificate Authority. Before using this form, you must copy the signed certificate into a file which you specify below.

Certificate store:	*SYSTEM	
Signed certificate path and file name:	/verisign/certificates/vscert3.txt	(required)

Figure 93. Receiving a System Certificate Issued by an Internet CA

5. Complete the Receive a System Certificate page (Figure 93) by typing the directory path and file name where you stored the signed system certificate received for the Internet CA. Click **OK**. The Certificate Received page as shown in Figure 94 is displayed.



Digital Certificate Manager

Certificate Received

Message The signed system certificate was successfully received into the certificate store.

File name: *SYSTEM

Use the Work with secure applications task if you want to specify that applications use this certificate.

Figure 94. Confirmation of Successful Receipt

6. Click **OK**. You will return to the Receive a System Certificate page (Figure 93). Click **Cancel**. You now need to specify which applications will use this system certificate. Refer to 9.5.5, “Configuring Web Server to Use SSL Server Authentication (V4R4)” on page 135, for a sample procedure.

Chapter 10. Setting Up the Network

This chapter contains the information that you must know to implement the infrastructure for your Net.Commerce. Before you implement your Net.Commerce solution, you must carefully plan how you are going to connect to the Internet, protect your recourses, connect your Net.Commerce server with your back-end system. Plus, you must install all of the necessary program products. For planning information, please read Chapter 2, “Planning: The Infrastructure” on page 9, before you continue.

10.1 Security

It is important that your implementation follow your company security policy. Here are some examples from different parts of a security policy.

10.1.1 General I/T Security Policy Statement

Normally, an I/T security policy has several pages. What we provide here are only some small parts from an example I/T security policy. In the following example, we call the company *Mycompany*.

- A Mycompany Information System (IS) is any automated information or telecommunications system owned, leased, or operated by Mycompany.
- Mycompany will implement at least the minimum security requirements as identified in this policy, to protect IS resources and information (non-sensitive and sensitive data) processed, stored, or transmitted by Mycompany ISs. Based on risk management, they may apply additional safeguards to provide the most restrictive set of controls (privileges) that permit the performance of authorized tasks (principle of least-privilege).
- Sensitive information in Mycompany ISs must be safeguarded against unauthorized disclosure, modification, access, use, destruction, or delay in service.
- All ISs processing, storing, or transmitting sensitive information must be accredited.
- Connectivity is prohibited between Mycompany IS and any other systems or networks not under Mycompany authority, unless formally approved by an appropriate Company Accrediting Authority.
- All Mycompany ISs are for Mycompany business only and users have no expectation of privacy while using these resources.
- All persons who use, manage, operate, maintain, or develop Mycompany ISs, applications, or data must comply with these policies.

10.1.2 Internet Services Policy

Mycompany owned or controlled ISs may only access the Internet through Mycompany approved gateways.

This limitation means that Mycompany owned, controlled, or authorized computer equipment, regardless of its location or means of connection to any network or system, may not be used to access the Internet, directly or indirectly. The only way is if the connection is through a Mycompany-approved Internet gateway (firewall). While the configuration of some networks make it technically possible to access the Internet without going through an approved gateway, such access is not authorized.

Exceptions to this policy must be approved in writing by the Director of the Telecommunications Department.

10.2 Server Placement

When implementing a Net.Commerce solution, you have to follow your company's security policy. In our example, we only allow a connection to the Internet through a firewall. Since we must protect our data on both the back-end system and the Net.Commerce server, we have to protect it with a firewall.

We use private IP-addresses on our example network. Private IP-addresses cannot be used on the Internet. Internet routers do not route private IP-addresses. The easiest way to make our Net.Commerce server on our private network visible on the Internet is to use Network Address Translation (NAT) in the firewall. We are not going to let our Net.Commerce server be directly attached to the Internet.

10.2.1 Scenario Objectives

The objectives of this scenario are to:

- Allow internet clients to access the Net.Commerce server
- Allow the back-end system to update the Net.Commerce server database

10.3 Firewall

Since we must protect our data on the Net.Commerce server and on our back-end system, we are going to implement a firewall solution. In our example, we are going to use the Firewall for AS/400 to protect us and our network.

Because the Net.Commerce server is a public server and accessed from the Internet, it needs a public address. We are going to use NAT to map a private address to a public address, for use over the Internet. We are going to use the non-secure port of the firewall as the public address for the Net.Commerce server.

This section describes the tasks that you must perform to install and configure a firewall using NAT. Figure 95 on page 151 shows our network configuration for this scenario.

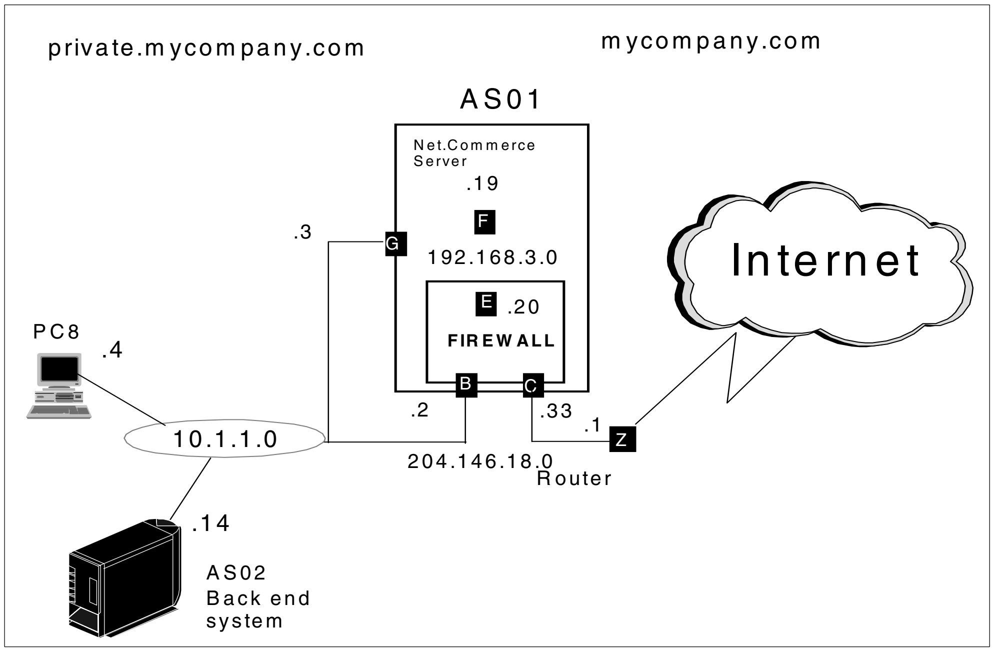


Figure 95. Scenario Network Configuration

Our scenario configuration includes two AS/400 servers in the mycompany.com network. AS01 houses the firewall, as well as a Net.Commerce server, behind the firewall. Internet clients access the public Web server by using the same IP address as the non-secure port of the firewall, 204.146.18.33. The Web server IP address is actually the AS/400 system **INTERNAL* port IP address 192.168.3.19 (**F**). You can use NAT to map the private address to the public one. AS02 is the back-end system in the secure network.

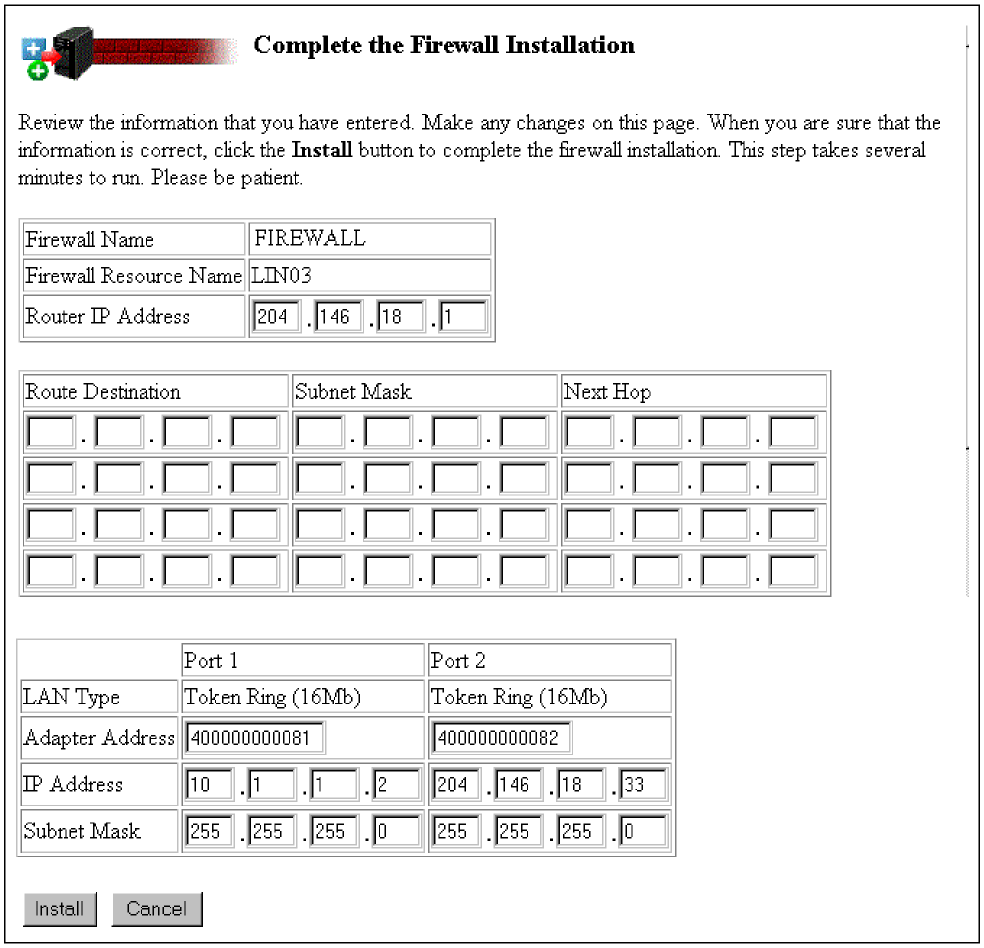
10.3.1 Task Summary

The following is a summary of tasks used to implement this NAT environment:

1. Install the firewall and start it successfully.
2. Perform basic configuration for the local firewall. Select the services that you want your internal users to have on the Internet (HTTP and mail, for example). Select a public HTTP server behind the firewall. In our scenario, we do not allow internal users to connect to the Internet.
3. Start NAT.
4. Add a default route to AS01 TCP/IP configuration pointing to the **INTERNAL* port of the firewall as the next hop to enable responses from the HTTP server behind the firewall to the Internet clients.
5. Restart the filters.
6. Verify the following items:
   - An Internet user can open a Web page on the Net.Commerce server behind the firewall.
   - Internal clients can use SOCKS and Proxy to open a Web page on the Internet (this is optional).

10.3.2 Installing the AS/400 Firewall

Install the firewall at the local site using the instructions in the manual *Getting Started with IBM Firewall for AS/400,* SC41-5424. A summary of the installation parameters is shown on the Complete the Firewall Installation summary page in Figure 96.



Complete the Firewall Installation

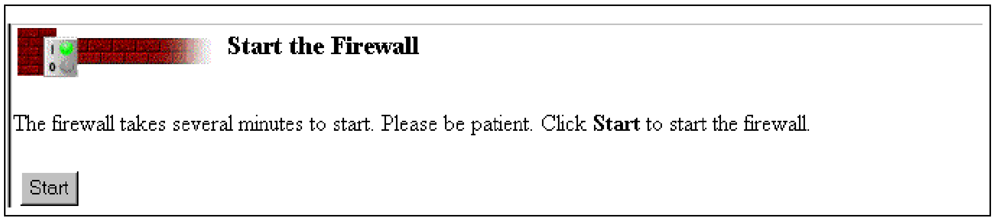
Review the information that you have entered. Make any changes on this page. When you are sure that the information is correct, click the **Install** button to complete the firewall installation. This step takes several minutes to run. Please be patient.

Firewall Name	FIREWALL
Firewall Resource Name	LIN03
Router IP Address	204 . 146 . 18 . 1

Route Destination	Subnet Mask	Next Hop
. . .	. . .	. . .
. . .	. . .	. . .
. . .	. . .	. . .
. . .	. . .	. . .

	Port 1	Port 2
LAN Type	Token Ring (16Mb)	Token Ring (16Mb)
Adapter Address	400000000081	400000000082
IP Address	10 . 1 . 1 . 2	204 . 146 . 18 . 33
Subnet Mask	255 . 255 . 255 . 0	255 . 255 . 255 . 0

Figure 96. Firewall Installation Summary Page



Start the Firewall

The firewall takes several minutes to start. Please be patient. Click **Start** to start the firewall.

Figure 97. Starting the Firewall

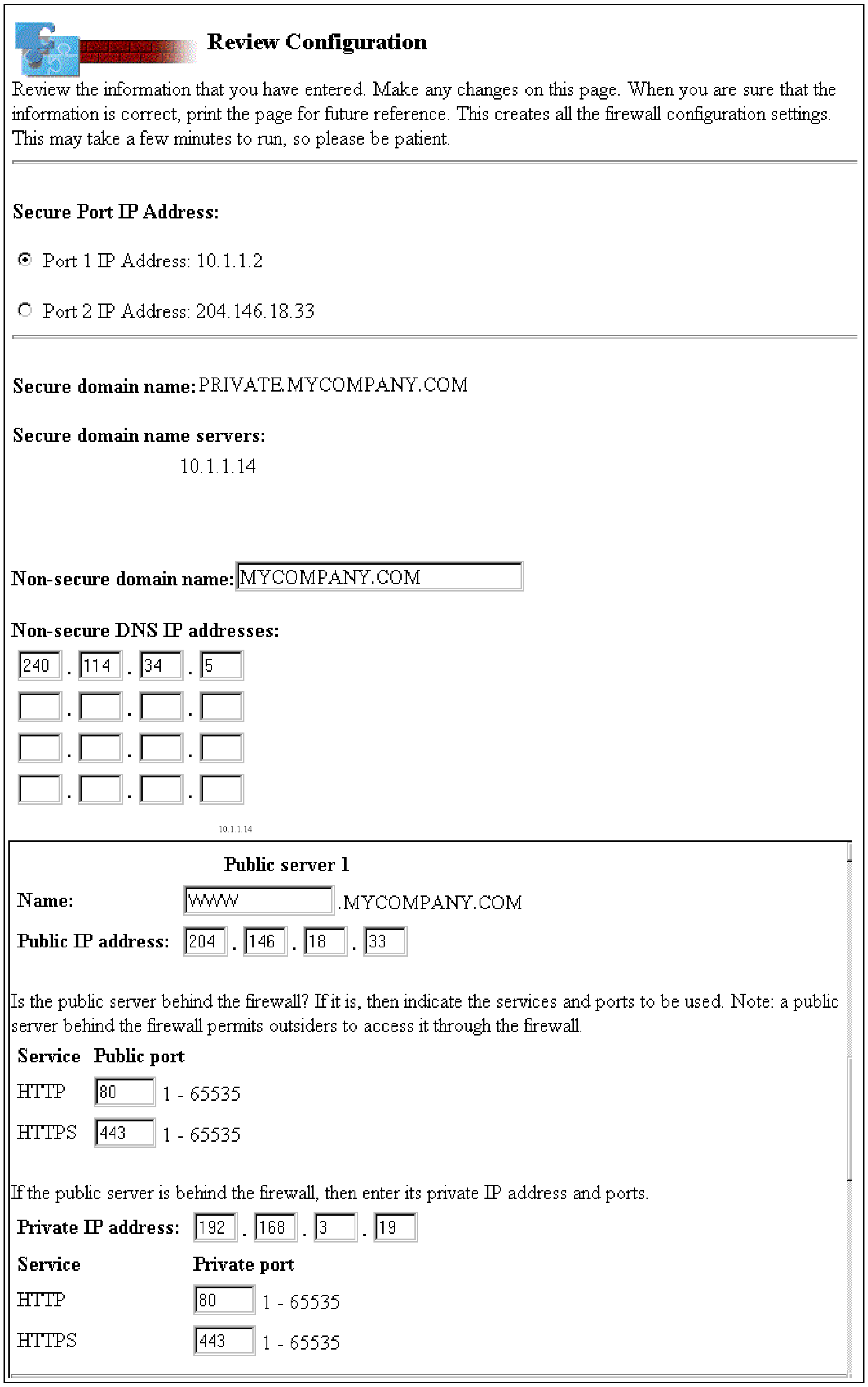
Start the firewall (Figure 97) by clicking **Start**.

10.3.3 Performing Basic Configuration

Perform the basic configuration of the local firewall (FIREWALL). For further information, refer to *Getting Started with IBM Firewall for AS/400,* SC41-5424, and the redbooks *AS/400 Internet Security: IBM Firewall for AS/400,* SG24-2162, and *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376.

The Review Configuration page shown in Figure 98 on page 154 and Figure 99 on page 155 show our configuration on the local system, AS01. Refer to Figure 95 on page 151 for the scenario network configuration.

Notice that we entered the name of the Net.Commerce server and its *public* IP address. In this example, the public IP address of the Net.Commerce server is the same as the non-secure port of the firewall. The next section of the page asks if the public server is behind the firewall. If so, enter the public ports that will be used for HTTP and HTTPS. We selected the well-known ports of 80 and 443 for HTTP and HTTPS, respectively. We also entered the private IP address of the Net.Commerce server, which is the home AS/400 **INTERNAL* port (**F** in Figure 95 on page 151) of the firewall. Remember that the Net.Commerce server is on the same AS/400 system that houses the firewall. Information about the Net.Commerce server is used to automatically generate the appropriate NAT settings and filter rules for accessing the Net.Commerce server behind the firewall.



Review Configuration

Review the information that you have entered. Make any changes on this page. When you are sure that the information is correct, print the page for future reference. This creates all the firewall configuration settings. This may take a few minutes to run, so please be patient.

Secure Port IP Address:

● Port 1 IP Address: 10.1.1.2

○ Port 2 IP Address: 204.146.18.33

Secure domain name: PRIVATE.MYCOMPANY.COM

Secure domain name servers:

10.1.1.14

Non-secure domain name: MYCOMPANY.COM

Non-secure DNS IP addresses:

240 . 114 . 34 . 5

. . .

. . .

. . .

10.1.1.14

Public server 1

Name: WWW .MYCOMPANY.COM

Public IP address: 204 . 146 . 18 . 33

Is the public server behind the firewall? If it is, then indicate the services and ports to be used. Note: a public server behind the firewall permits outsiders to access it through the firewall.

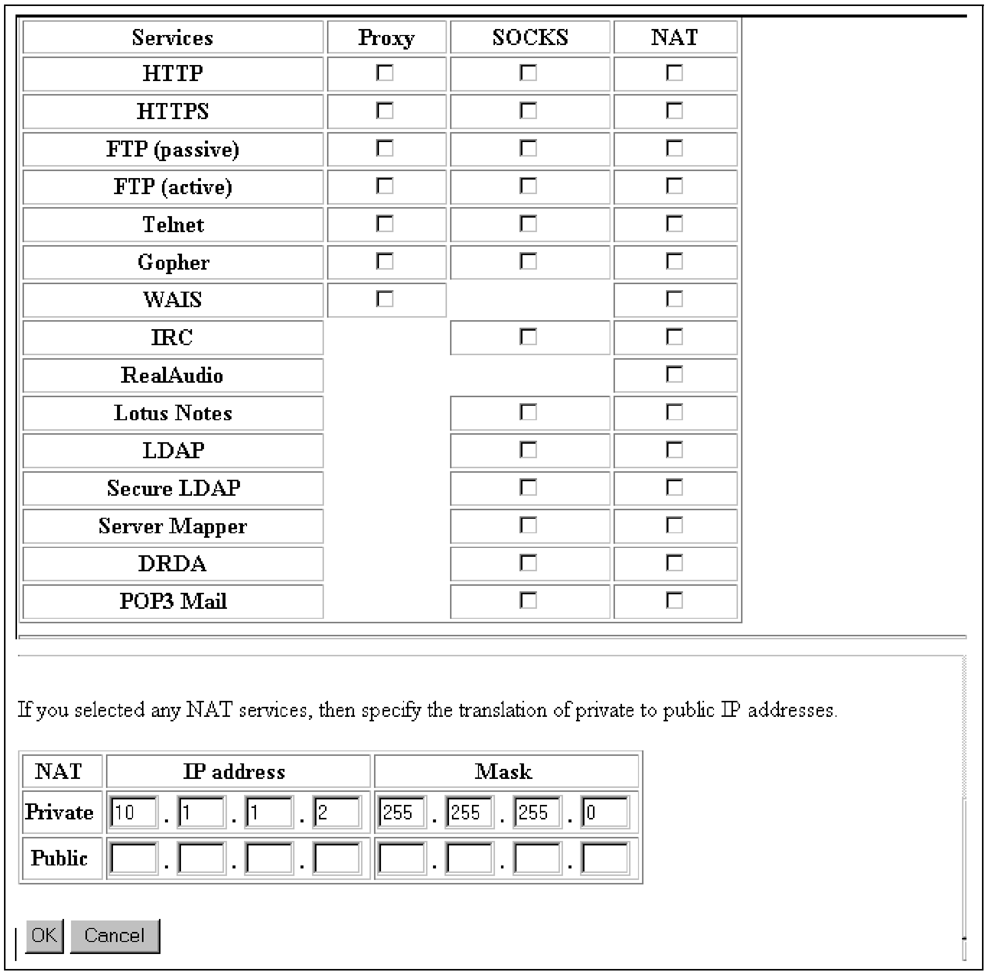
Service	Public port	
HTTP	80	1 - 65535
HTTPS	443	1 - 65535

If the public server is behind the firewall, then enter its private IP address and ports.

Private IP address: 192 . 168 . 3 . 19

Service	Private port	
HTTP	80	1 - 65535
HTTPS	443	1 - 65535

Figure 98. Firewall Basic Configuration Summary Page (Part 1 of 2)



Services	Proxy	SOCKS	NAT
HTTP	☐	☐	☐
HTTPS	☐	☐	☐
FTP (passive)	☐	☐	☐
FTP (active)	☐	☐	☐
Telnet	☐	☐	☐
Gopher	☐	☐	☐
WAIS	☐		☐
IRC		☐	☐
RealAudio			☐
Lotus Notes		☐	☐
LDAP		☐	☐
Secure LDAP		☐	☐
Server Mapper		☐	☐
DRDA		☐	☐
POP3 Mail		☐	☐

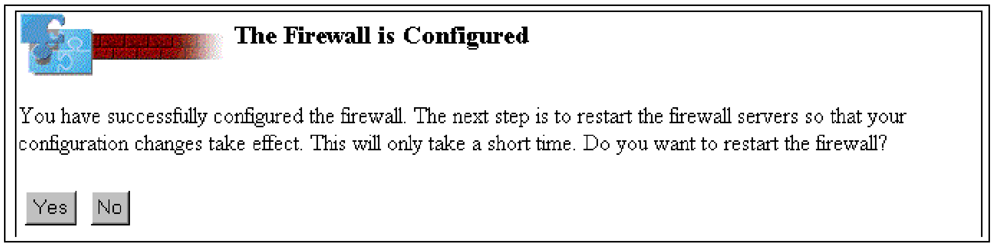
If you selected any NAT services, then specify the translation of private to public IP addresses.

NAT	IP address	Mask
Private	10 . 1 . 1 . 2	255 . 255 . 255 . 0
Public	. . .	. . .

Figure 99. Firewall Basic Configuration Summary Page (Part 2 of 2)

Complete the following steps:

1. Click **OK**. A confirmation page (Figure 100) is shown. It indicates that the firewall is configured. It is not necessary to restart the firewall at this time because we have more configuration work to do.



The Firewall is Configured

You have successfully configured the firewall. The next step is to restart the firewall servers so that your configuration changes take effect. This will only take a short time. Do you want to restart the firewall?

Figure 100. Confirmation that the Firewall Is Configured

2. Click **No**.

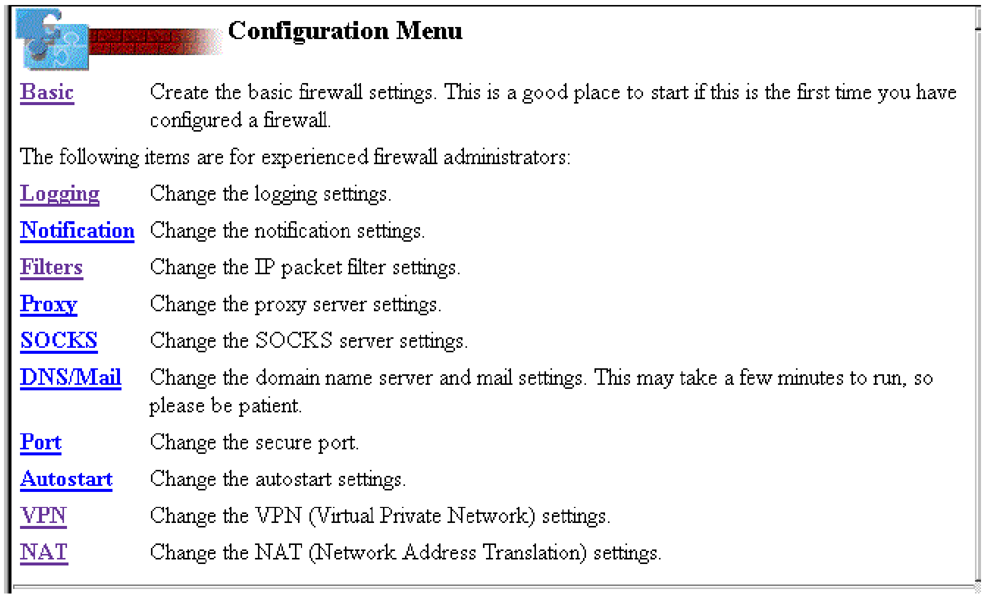
10.3.4 Changing NAT Rules

Basic configuration automatically creates the NAT filter rules to allow HTTP and HTTPS traffic. However, if you want to use Secure Electronic Transaction (SET), you must create the additional rules that we have to communicate with eWallet on the shopper's PC and with the Payment Gateway. The default ports for eWallet communication is 8614, 8620, XXX and for the Gateway Server 8888. If your SET certifying authority is using some special port for SET certificate, you have to consider that too.

For further information about NAT, refer to *Getting Started with IBM Firewall for AS/400,* SC41-5424, and the redbook *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376.

For information regarding SET, refer to the site on the Web at:
`http://www.setco.org`

In this example, we add the eWallet rule (port 8614) to the NAT. You have to add a rule for each port that you want to use on your server. We also change the NAT filter rules to translate all ports.



Configuration Menu

Basic	Create the basic firewall settings. This is a good place to start if this is the first time you have configured a firewall.

The following items are for experienced firewall administrators:

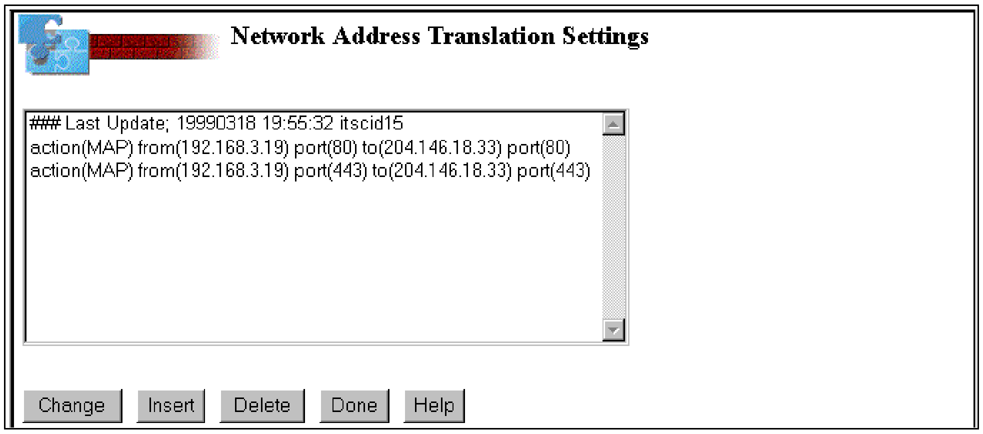
Logging	Change the logging settings.
Notification	Change the notification settings.
Filters	Change the IP packet filter settings.
Proxy	Change the proxy server settings.
SOCKS	Change the SOCKS server settings.
DNS/Mail	Change the domain name server and mail settings. This may take a few minutes to run, so please be patient.
Port	Change the secure port.
Autostart	Change the autostart settings.
VPN	Change the VPN (Virtual Private Network) settings.
NAT	Change the NAT (Network Address Translation) settings.

Figure 101. Selection of NAT from the Configuration Menu

To begin, perform the following steps:

1. Click **NAT** on the Configuration Menu page (Figure 101).

   The Network Address Translation Settings page appears as shown in Figure 102 on page 157. Notice that IBM Firewall for AS/400 already generated two MAP settings for us. They are based on the information we provided in the Public server 1 section of Basic configuration (Figure 98 on page 154).



```
### Last Update; 19990318 19:55:32 itscid15
action(MAP) from(192.168.3.19) port(80) to(204.146.18.33) port(80)
action(MAP) from(192.168.3.19) port(443) to(204.146.18.33) port(443)
```

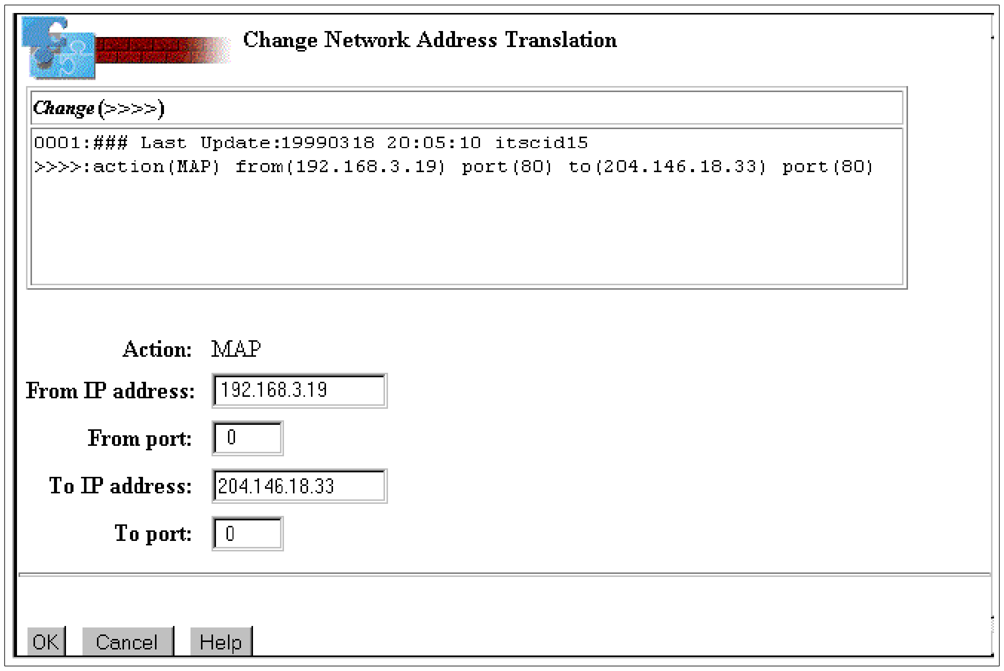
Figure 102. Network Address Translation Settings Page

2. Select the last MAP setting in the list (Figure 102). Click **Delete**.
3. Select the remaining MAP setting in the list (Figure 102). Click **Change**. The Change Network Address Translation page is shown (Figure 103 on page 158).

   Figure 103 shows the Change Network Address Translation page. Change the *From port* to "0" and the *To port* to "0". Port 0 tell the NAT translation to pass all communication on all ports. Changing the NAT port to 0 makes us depend on the filter rules for protection of the Net.Commerce server. Therefore, it is very important that you only allow the IP-packets that you want to get through the firewall filters. During the SET payment, the Payment Server uses randomly selected ports above 1023 to communicate to the Payment Gateway.

Tip

Remember that the *From* port is always the secure (hidden) address and the *To* address is the registered address that you want to publish.

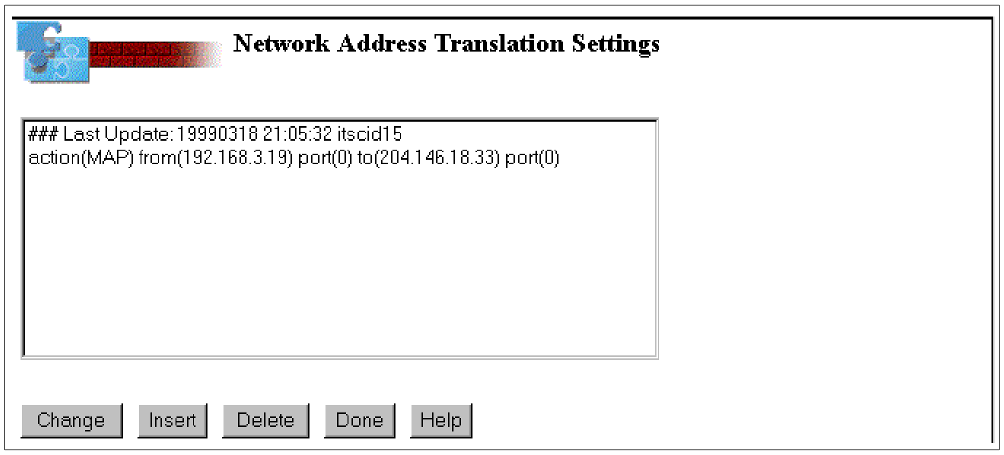


```
0001:### Last Update:19990318 20:05:10 itscid15
>>>>:action(MAP) from(192.168.3.19) port(80) to(204.146.18.33) port(80)
```

Action:	MAP
From IP address:	192.168.3.19
From port:	0
To IP address:	204.146.18.33
To port:	0

Figure 103. Change the NAT MAP Setting

4. Because the *From IP address* is always the secure (hidden) address, in our environment, this is 192.168.3.19, and the port to map is 0 (all port). The *To* IP address is the address that we want to publish, which is 204.146.18.33 (the non-secure port of the firewall), also using port 0 (all port). After entering the required information, click **OK** to continue.



```
### Last Update:19990318 21:05:32 itscid15
action(MAP) from(192.168.3.19) port(0) to(204.146.18.33) port(0)
```

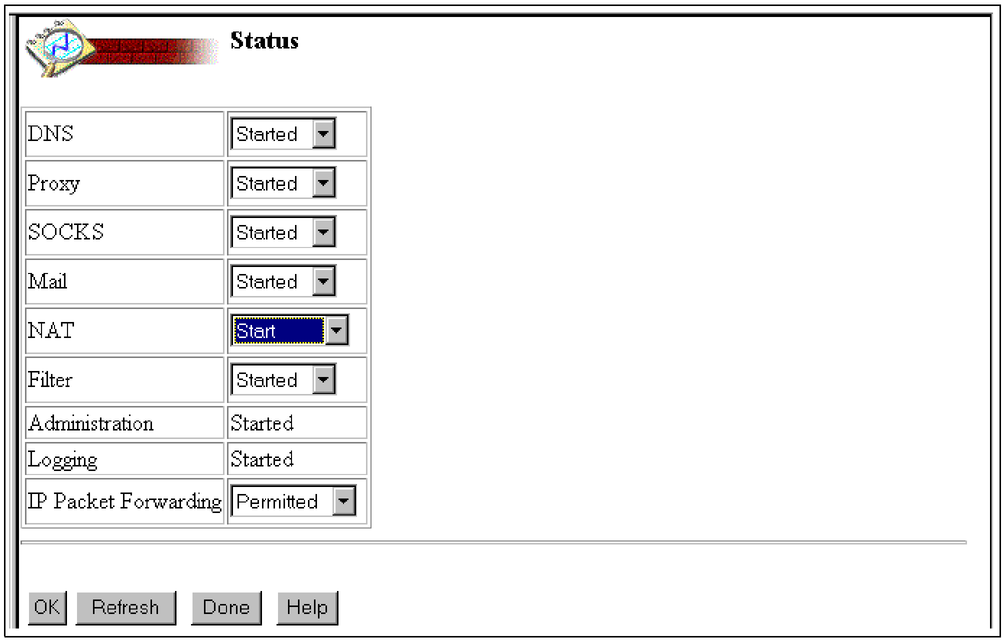
Figure 104. Displaying NAT Settings

5. The resulting NAT setting is shown for confirmation (see Figure 104). If you have more settings to add, you can do so now. In this scenario, this is the only NAT setting we need to add. Click **Done**.

   You are returned to the Firewall Installation Tasks page.

10.3.5 Starting NAT

Click the **Administration** icon. Then, click **Status** from the Administration Menu page. Start NAT as shown in Figure 105.


Status

DNS	Started
Proxy	Started
SOCKS	Started
Mail	Started
NAT	Start
Filter	Started
Administration	Started
Logging	Started
IP Packet Forwarding	Permitted

Figure 105. Starting NAT from the Status Page

10.3.6 Adding Filter Rules for SET

Basic configuration automatically creates the filter rules to allow HTTP and HTTPS traffic. However, you must create the additional rules for SET. We have to communicate with the eWallet and with the Payment Gateway. The default port for eWallet is 8614, 8620, XXXX and to Gateway Server 8888. You must be sure that you do not override the rules that Basic configuration created and to make it easier to recognize rules that you manually add after the initial configuration of the firewall. We recommend that you create a section at the bottom of the filter rules just before the *Ending defense* section. Enter a title such as *Custom Rules*.

Important

Always test your filter rules before you use them. One simple misspelling can open your filter.

10.3.7 Filter Rules for Requesting a Certificate

You need to add the rules listed in this section to allow the Payment Server to request a digital SET certificate from a certify authority that uses port 5065 for a certificate request. If your certificate authority is using port 80 or 443 for certificate requests, you do not need to add any filter rules for the certificate request. Those filters were created when you selected to have the public server behind the firewall, during the basic configuration of the firewall. See Figure 98 on page 154.

Add the following four filter rules to your firewall:

```
permit 192.168.3.19 255.255.255.255 0.0.0.0 0.0.0.0 tcp ge 1024 eq 5065 secure
route inbound f=y l=y t=0 # Permit SET Cert. request 1/2

permit 192.168.3.19 255.255.255.255 0.0.0.0 0.0.0.0 tcp ge 1024 eq 5065
non-secure route outbound f=y l=n t=0 # Permit SET Cert. request 2/2

permit 0.0.0.0 0.0.0.0 204.146.18.33 255.255.255.255 tcp eq 5065 ge 1024
non-secure both inbound f=y l=n t=0 # Permit SET Cert. request response 1/2

permit 0.0.0.0 0.0.0.0 192.168.3.19 255.255.255.255 tcp eq 5065 ge 1024 secure
both outbound f=y l=n t=0 # Permit SET Cert. request response 2/2
```

You should consider to remove or change these filter rules to deny when you have received your digital SET certificate.

For more information about adding and implementing filter rules, please refer to *IBM Internet Security: IBM Firewall for AS/400*, SG24-2162.

10.3.8 Setting Up SOCKS for a Certificate Request

If you are going to request a digital SET certificate from VeriSign, you have to configure SOCKS client support on your AS/400 system. You have to use Operations Navigator to configure the SOCKS on your AS/400 system.

Go to **TCP/IP Properties** and select the **SOCKS** tab. The SOCKS information window appears (Figure 106).

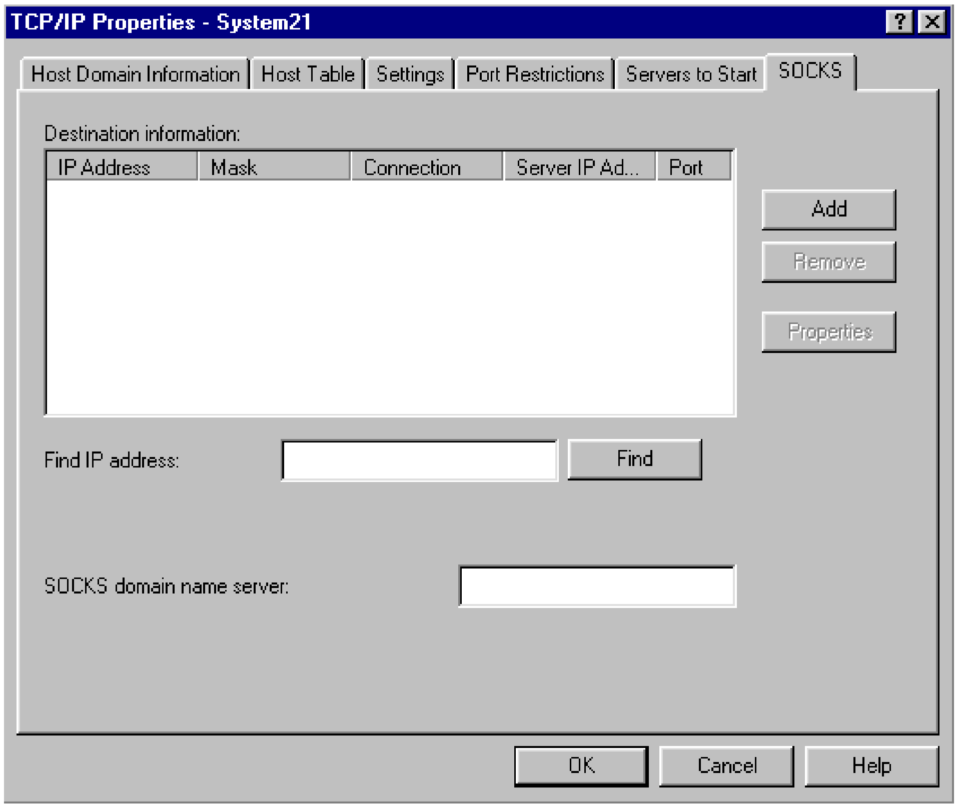


Figure 106. Operations Navigator — TCP/IP Properties SOCKS before Configuration

You are now ready to configure SOCKS information for your AS/400 system. To configure the SOCKS information for the AS/400 system, you must provide at least two pieces of information:

- The network that is directly connected to the AS/400 system. A SOCKS server is not needed to reach the network.
- The network that requires the use of a SOCKS server for access and the SOCKS server to use to access the network.

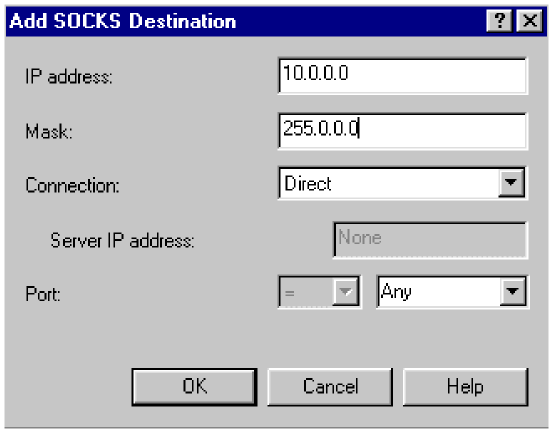
As an option, you can add a DNS server to be used by SOCKS.

10.3.8.1 Defining the Direct Network

Do not use the SOCKS server to connect to any network that is directly connected to the system. To prevent the AS/400 system from connecting through the SOCKS server, the directly connected network should be defined.

To define the directly connected network, complete these steps:

1. In the SOCKS information window, click the **Add** button. The Add SOCKS Destination window appears (Figure 107).


Add SOCKS Destination

IP address: 10.0.0.0

Mask: 255.0.0.0

Connection: Direct

Server IP address: None

Port: = Any

Figure 107. Add SOCKS Destination with Direct Connection Information

2. Type the network address of the secure network in the IP address field. In our sample network, we use 10.0.0.0.
3. Type the subnet mask that describes your secure network in the Mask field. In our sample network, we use a subnet mask of 255.0.0.0.
4. Click the down arrow in the Connection field, and select **Direct** from the list of options.
5. Click **OK** to add the destination information.

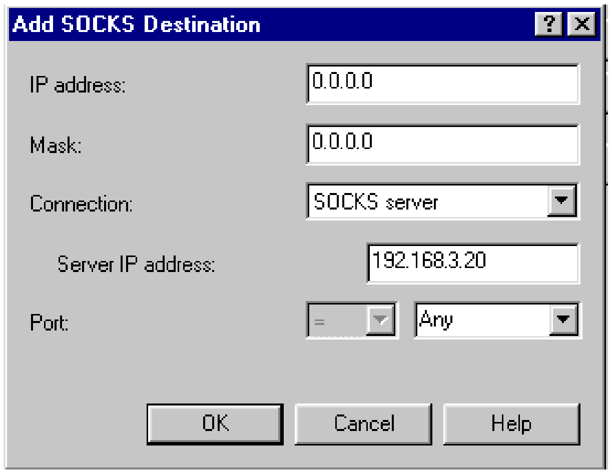
You have now defined the “10.” network as a direct network. SOCKS does not access any host with an address that starts with “10.”

10.3.8.2 Defining the Network Connection Using SOCKS

Now, you must define the network to use with the SOCKS server. In this example, we use the SOCKS server to access all networks except the direct connection.

To define the network for use with SOCKS, follow these steps:

1. In the SOCKS information window, click the **Add** button. The Add SOCKS Destination window appears (Figure 108).



IP address: 0.0.0.0
Mask: 0.0.0.0
Connection: SOCKS server
Server IP address: 192.168.3.20
Port: = Any

Figure 108. Add SOCKS Destination with SOCKS Server Connection

2. Type the address `0.0.0.0` in the IP address field.
3. Type the subnet mask `0.0.0.0` in the Mask field.

   When a destination address is “ANDed” with a mask of 0.0.0.0, the result is 0.0.0.0. By specifying a mask and address of all zeros, all IP addresses match this destination description.
4. Click the down arrow in the Connection field, and select **SOCKS Server** from the list of options.
5. Type the IP address of the SOCKS server in the Server IP Address field. On the AS/400 system with the firewall installed, this is the IP address of the *INTERNAL port of the firewall. On other AS/400 systems in the secure network, this is the IP address of the secure port of the firewall.
6. Verify that the Port field is set to "Any". This specifies the remote ports for which this connection can be used.
7. Click **OK** to add the destination information.

You have now defined the destination information for SOCKS. You may also need to configure the SOCKS domain name server.

10.3.8.3 Defining the SOCKS Domain Name Server

The SOCKS domain name server field specifies the IP address of a DNS server that can resolve names or IP addresses that reside on a non-secure network. Leave this field blank if the domain name servers configured with TCP/IP resolve the addresses.

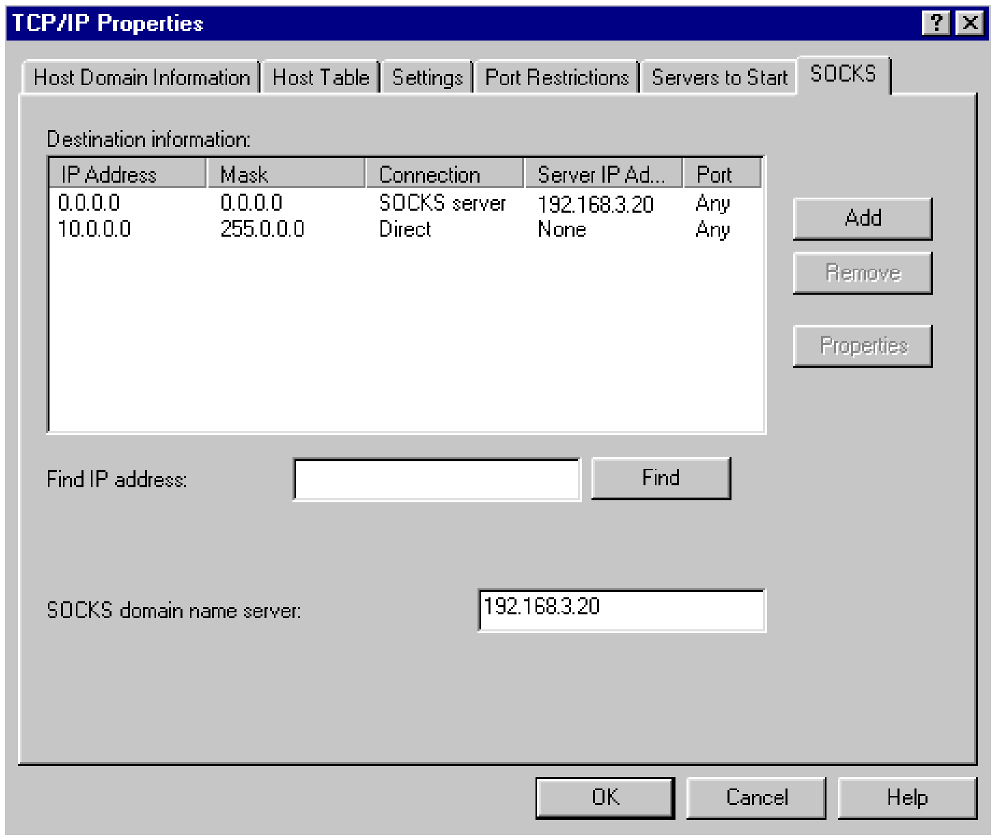
For name or IP address resolution, the system queries the DNS servers configured with TCP/IP first. If they cannot resolve the name or address, the system queries the DNS server that you specify.

Note

At least one DNS server must be configured by using CFGTCP option 12 before SOCKS checks the domain name server configured for SOCKS.

If you do not have an internal DNS server, point the AS/400 system at the firewall for DNS services. If the internal DNS server cannot resolve external information, type the IP address of the firewall in the SOCKS domain name server field. On the AS/400 system with the firewall installed, this is the IP address of the *INTERNAL port of the firewall. On the other AS/400 systems in the secure network, this is the IP address of the secure port of the firewall.

After you enter all of your SOCKS information, your SOCKS information window should appear similar to the one shown in Figure 109. Click **OK** to save the configuration. The Operations Navigator window appears.



Destination information:

IP Address	Mask	Connection	Server IP Ad...	Port
0.0.0.0	0.0.0.0	SOCKS server	192.168.3.20	Any
10.0.0.0	255.0.0.0	Direct	None	Any

Find IP address:

SOCKS domain name server: 192.168.3.20

Figure 109. Point to the SOCKS Domain Name Server

10.3.9 Filter Rules for SET Communication

You need to add the following rules to allow the Card Holder Requests to and from the eWallet, Pay Authorizations and Payment Capture to the Payment Gateway. Here are the filter rules to allow communication between the Payment Server and the eWallet. In our example, it is port 8620:

```
permit 0.0.0.0 0.0.0.0 204.146.18.33 255.255.255.255 tcp ge 1024 eq 8620
non-secure both inbound f=y l=y t=0

permit 0.0.0.0 0.0.0.0 192.168.3.19 255.255.255.255 tcp ge 1024 eq 8620
secure route outbound f=y l=y t=0

permit 192.168.3.19 255.255.255.255 0.0.0.0 0.0.0.0 tcp eq 8620 ge 1024
secure route inbound f=y l=y t=0

permit 192.168.3.19 255.255.255.255 0.0.0.0 0.0.0.0 tcp eq 8620 ge 1024
non-secure route outbound f=y l=y t=0
```

Here are the filter rules to allow communication between the Payment Server and the Payment Gateway. In our example, it is port 10010. The default port for communication with an acquirer (Payment Gateway) is 8888.

```
permit 192.168.3.19 255.255.255.255 0.0.0.0 0.0.0.0 tcp ge 1024 eq 10010
secure route inbound f=y l=y t=0

permit 192.168.3.19 255.255.255.255 0.0.0.0 0.0.0.0 tcp ge 1024 eq 10010
non-secure route outbound f=y l=y t=0

permit 0.0.0.0 0.0.0.0 204.146.18.33 255.255.255.255 tcp eq 10010 ge 1024
non-secure both inbound f=y l=y t=0

permit 0.0.0.0 0.0.0.0 192.168.3.19 255.255.255.255 tcp eq 10010 ge 1024
secure route outbound f=y l=y t=0
```

Important

These filter rules may not apply to your system. The ports used by your acquirer may differ from the ports used in our example.

For more information about how to add and implement filter rules, please refer to *IBM Internet Security: IBM Firewall for AS/400*, SG24-2162.

10.3.10 Configuring a Default Route to Route Web Server Responses

Because you have the Net.Commerce server on the same AS/400 system that houses the firewall (AS01 in our scenario), you must add a default route that specifies the **INTERNAL* IP address of the firewall (interface **E**, Figure 95 on page 151) as the next hop. This allows the Internet clients to receive responses from the server (which must be routed through the firewall). Refer to Figure 111 on page 165 for an example of the default route configuration on AS01 entry.

10.3.11 Restarting the Filters

To restart the filters, click the firewall **Administration** icon, and then click **Status** from the Administration Menu page. Select **Restart** for the filters and click **OK**. Refer to Figure 105 on page 159 for an example of the Status page.

10.3.12 Verifying Access to the Web Server and Internet

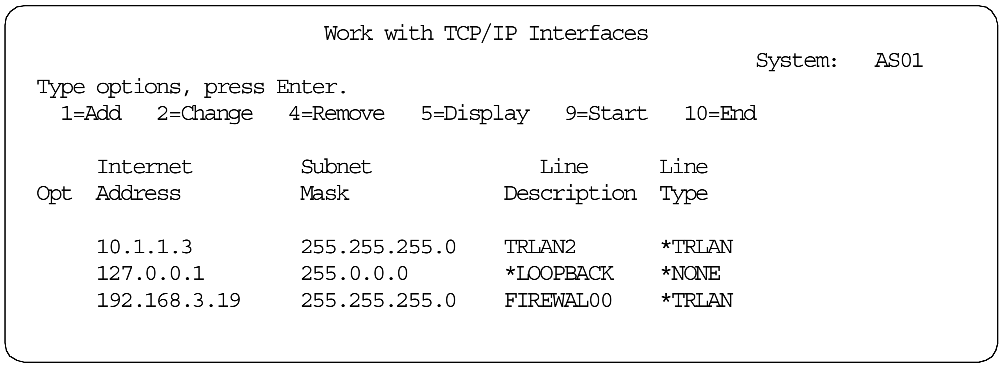
After completing the steps in our scenario, we performed the following verification testing. We successfully opened a Web page:

- On the Net.Commerce server behind the firewall (AS01) from the Internet.
- On the Internet from an internal client in the secure network using SOCKS as well as Proxy (this is optional).

If you want to have more information about filter rules and NAT, please refer to *IBM Internet Security: IBM Firewall for AS/400*, SG24-2162, and *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376.

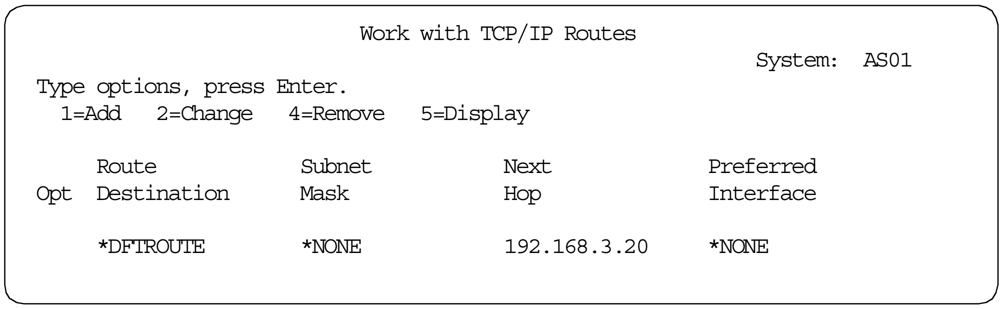
10.3.13 Additional Configuration Information

This section shows the TCP/IP configuration and network server descriptions for the firewall configuration FIREWALL on system AS01.



```
                          Work with TCP/IP Interfaces
                                                         System:   AS01
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display   9=Start   10=End

     Internet         Subnet              Line       Line
Opt  Address          Mask             Description   Type

     10.1.1.3         255.255.255.0    TRLAN2        *TRLAN
     127.0.0.1        255.0.0.0        *LOOPBACK     *NONE
     192.168.3.19     255.255.255.0    FIREWAL00     *TRLAN
```

Figure 110. AS/400 System TCP/IP Interfaces



```
                             Work with TCP/IP Routes
                                                         System:  AS01
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display

     Route            Subnet           Next             Preferred
Opt  Destination      Mask             Hop              Interface

     *DFTROUTE        *NONE            192.168.3.20     *NONE
```

Figure 111. AS/400 System Routing Configuration

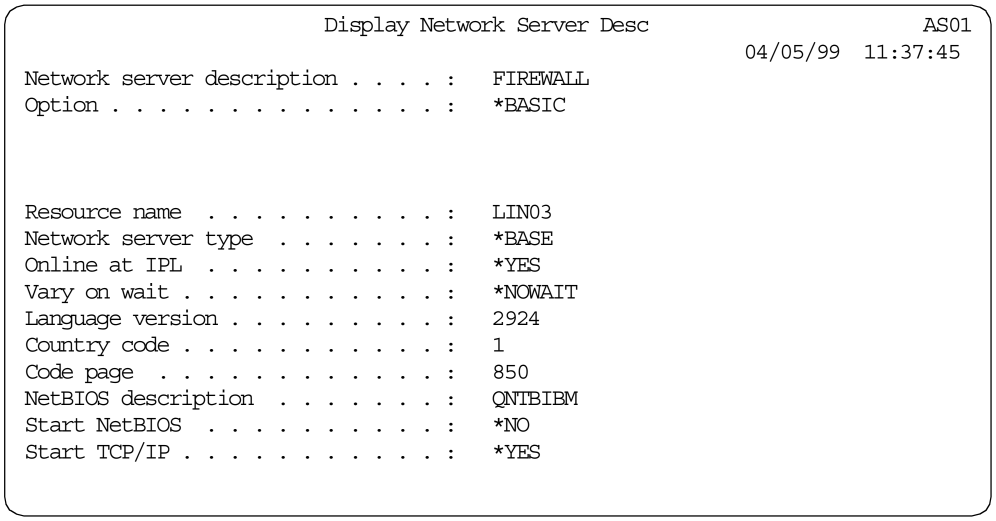

```
                          Display Network Server Desc                       AS01
                                                              04/05/99  11:37:45
Network server description . . . . :   FIREWALL
Option . . . . . . . . . . . . . . :   *BASIC



Resource name  . . . . . . . . . . :   LIN03
Network server type  . . . . . . . :   *BASE
Online at IPL  . . . . . . . . . . :   *YES
Vary on wait . . . . . . . . . . . :   *NOWAIT
Language version . . . . . . . . . :   2924
Country code . . . . . . . . . . . :   1
Code page  . . . . . . . . . . . . :   850
NetBIOS description  . . . . . . . :   QNTBIBM
Start NetBIOS  . . . . . . . . . . :   *NO
Start TCP/IP . . . . . . . . . . . :   *YES
```

Figure 112. Network Server Description (Part 1 of 7)

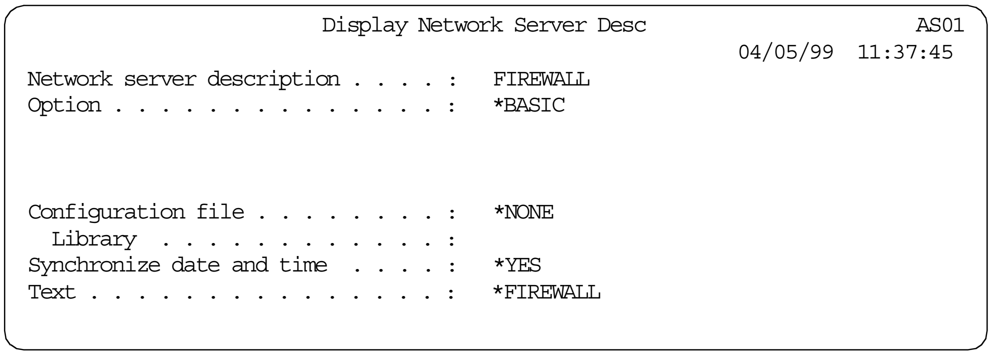

```
                          Display Network Server Desc                       AS01
                                                              04/05/99  11:37:45
Network server description . . . . :   FIREWALL
Option . . . . . . . . . . . . . . :   *BASIC



Configuration file . . . . . . . . :   *NONE
  Library  . . . . . . . . . . . . :
Synchronize date and time  . . . . :   *YES
Text . . . . . . . . . . . . . . . :   *FIREWALL
```

Figure 113. Network Server Description (Part 2 of 7)

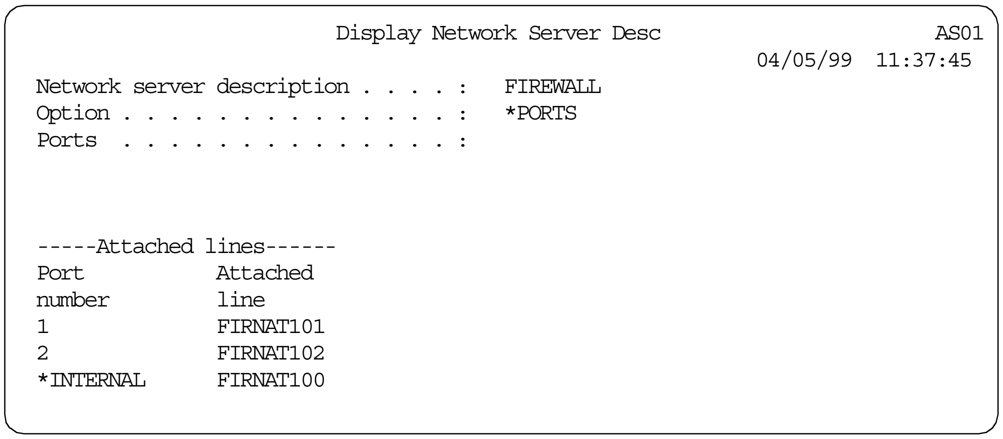

```
                          Display Network Server Desc                       AS01
                                                              04/05/99  11:37:45
Network server description . . . . :   FIREWALL
Option . . . . . . . . . . . . . . :   *PORTS
Ports  . . . . . . . . . . . . . . :



-----Attached lines------
Port          Attached
number        line
1             FIRNAT101
2             FIRNAT102
*INTERNAL     FIRNAT100
```

Figure 114. Network Server Description (Part 3 of 7)

```
                         Display Network Server Desc                      AS01
                                                             04/05/99  11:37:45
Network server description . . . . :   FIREWALL
Option . . . . . . . . . . . . . . :   *STGLNK
Storage space links  . . . . . . . :



---------------------------Storage space links---------------------------
Network
server
storage        Drive     Text
FIRNAT100       K
```

Figure 115. Network Server Description (Part 4 of 7)

```
                         Display Network Server Desc                      AS01
                                                             04/05/99  11:37:45
Network server description . . . . :   FIREWALL
Option . . . . . . . . . . . . . . :   *TCPIP
TCP/IP port configuration  . . . . :



--------------------TCP/IP port configuration--------------------
                                                         Maximum
               Internet            Subnet             transmission
Port           address             mask                   unit
1              10.1.1.2            255.255.255.0             1500
2              204.146.18.33       255.255.255.0             1500
*INTERNAL      192.168.3.20        255.255.255.0            15400
```

Figure 116. Network Server Description (Part 5 of 7)

```
                         Display Network Server Desc                      AS01
                                                             04/05/99  11:37:45
Network server description . . . . :   FIREWALL
Option . . . . . . . . . . . . . . :   *TCPIP
TCP/IP route configuration . . . . :



-------------TCP/IP route configuration--------------
Route               Subnet              Next
destination         mask                hop
*DFTROUTE           *NONE               204.146.18.1
```

Figure 117. Network Server Description (Part 6 of 7)

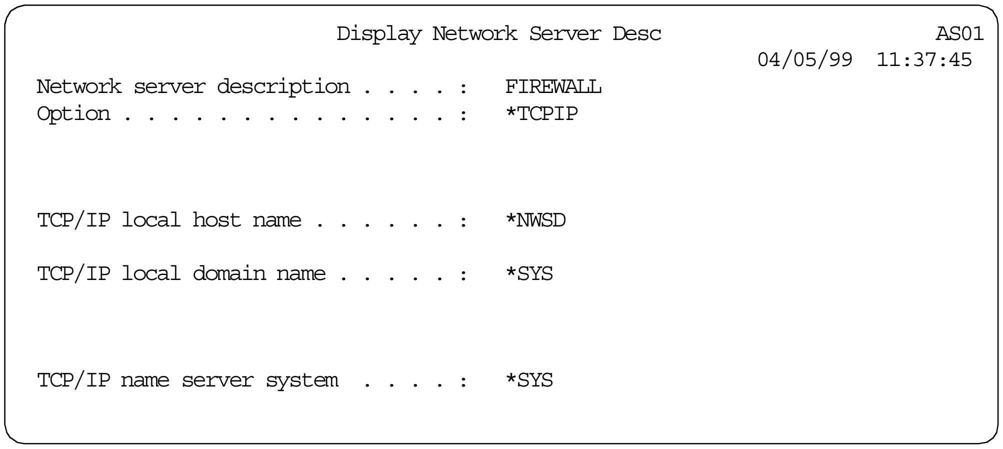

```
                         Display Network Server Desc                   AS01
                                                              04/05/99  11:37:45
 Network server description . . . . :   FIREWALL
 Option . . . . . . . . . . . . . . :   *TCPIP



 TCP/IP local host name . . . . . . :   *NWSD

 TCP/IP local domain name . . . . . :   *SYS



 TCP/IP name server system  . . . . :   *SYS
```

Figure 118. Network Server Description (Part 7 of 7)

For more detailed information about firewall implementation, please read the redbook *AS/400 Internet Security: IBM Firewall for AS/400*, SG24-2162. For detailed information about Network Address Translation (NAT), please read the redbook *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376.

10.3.14 OS/400 TCP/IP Configuration

Because you have a Net.Commerce server on the same AS/400 system that houses the firewall (AS01), you must add a default route specifying the *INTERNAL IP address of the firewall (interface **E** in Figure 95 on page 151) as the next hop. This allows Internet clients to receive responses from the server (which must be routed through the firewall). Refer to Figure 111 on page 165 for an example of the default route configuration on AS01.

There is no need to make manual changes to the firewall network server description for this scenario. Figure 119 and Figure 120 on page 169 show the TCP/IP interface and route configuration on system AS01.

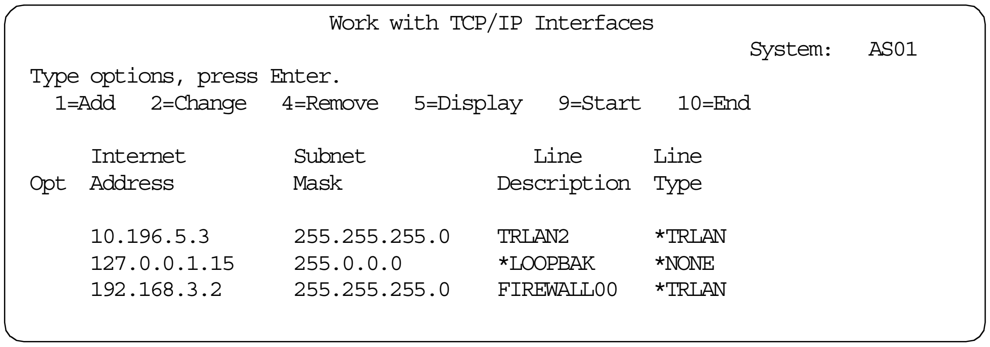
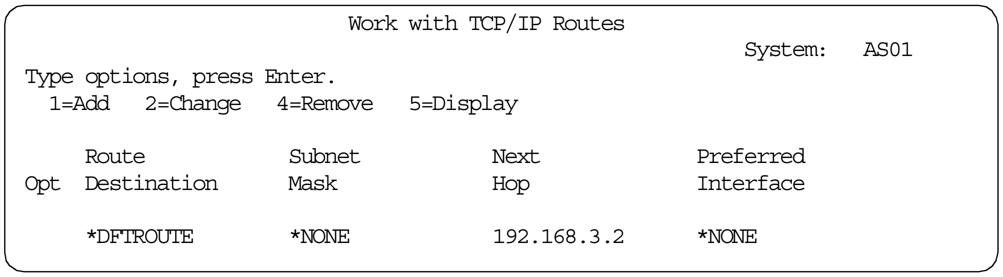

```
                         Work with TCP/IP Interfaces
                                                            System:   AS01
 Type options, press Enter.
   1=Add   2=Change   4=Remove   5=Display   9=Start   10=End

      Internet        Subnet            Line         Line
 Opt  Address         Mask              Description  Type

      10.196.5.3      255.255.255.0     TRLAN2       *TRLAN
      127.0.0.1.15    255.0.0.0         *LOOPBAK     *NONE
      192.168.3.2     255.255.255.0     FIREWALL00   *TRLAN
```

Figure 119. AS/400 System TCP/IP Interfaces — AS01



```
                           Work with TCP/IP Routes
                                                            System:   AS01
Type options, press Enter.
  1=Add   2=Change   4=Remove   5=Display

     Route            Subnet           Next             Preferred
Opt  Destination      Mask             Hop              Interface

     *DFTROUTE        *NONE            192.168.3.2      *NONE
```

Figure 120. AS/400 System Routing Configuration — AS01

10.4 Backend System Connection

As a part of your Net.Commerce implementation, you must have a connection between your Net.Commerce system and your backend system. We do not discuss security on the back-end system or any other security on the local network.

If you are interested in IP filtering on the AS/400 system, please read the redbook *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376.

To communicate between the Net.Commerce server and the back-end system, you must have some sort of communication. We assume that you have a Local Area Network (LAN) that connects your two systems.

Chapter 11. Installing Net.Commerce

This chapter describes how to perform pre-installation procedures and how to install the Net.Commerce on the AS/400 platform.

11.1 Pre-Installation Procedures

Before you install Net.Commerce, you must perform several pre-installation procedures. These procedures include determining the host name for your machine, determining or creating the relational database directory entry, and creating an AS/400 user ID.

To determine the unique fully qualified host name and domain name, use the `CFGTCP` command and select option **12** (change local domain).

To determine the relational database directory entry, go to the AS/400 command line and type `WRKRDBDIRE`. Press **PF4** to bring up the command prompt. Make a note of the directory entry name. If the directory does not exist, you must create one by selecting option **1** from the Work With Relational Database Directory Entries screen.

To create an AS/400 ID with QSECOFR authority, special authority *IOSYSCFG and *SECADM, perform the following steps:

1. Go to the AS/400 command line and type: `CRTUSRPRF`
2. Press **PF4** for a prompt.
3. Fill in the necessary parameters. Press **Enter** to create the user ID.

Please check and make sure you have the most current list of PTFs applied on your system. Use the `DSPPTF` command to display the PTFs on your machine and compare them with the latest PTF list.

11.2 Installing Net.Commerce

This section covers installation of all the components of the Net.Commerce on a single AS/400 system. For more information, please refer to *Net.Commerce for AS/400 Installing and Getting Started Guide* (ncinst.pdf), GC09-2864.

1. Refer to 2.2.1, “AS/400 Net.Commerce Software Requirements” on page 10, for a list of required software. Ensure that the required software and PTFs are installed.
2. Log on to your AS/400 system with the same user ID that you created in the previous section.
3. Insert the Net.Commerce CD-ROM into your AS/400 CD-ROM drive.
4. Type the `RSTLICPGM` command on the command line.
5. Press **PF4** for a prompt.
6. Type the product number (`5798NC3`) and device name in the appropriate entry fields.

   An acknowledgement message appears and indicates that you have installed Net.Commerce successfully.
7. Acquire and install any PTFs that are required for the newly installed products.

Chapter 12. Configuring Net.Commerce

This chapter describes how to configure the Net.Commerce instances using the Net.Commerce Configuration Manager. Specifically, if you want to implement the ITSO Redbook solution, you need to configure three Net.Commerce instances. The "Test" instance is for developing code. "Work" is for importing data after the development. "Product" is the production instance. For more information, such as a detailed explanation of setting options, what happens during configuration, and how the database collection is created, please refer to *Net.Commerce for AS/400 Installing and Getting Started Guide* (ncinst.pdf), GC09-2864.

Note

This redbook is based on Net.Commerce for AS/400 V3.2 running on OS/400 V4R3. While most of the redbook still applies to Net.Commerce V3.2 running on later versions of OS/400, some specific instructions are only for V4R3. Chapter 9, “Setting Up SSL Using DCM” on page 119, of this updated softcopy version has been modified to include DCM V4R4. If you are configuring or running Net.Commerce V3.2 on V4R4, please make sure you obtain Informational APAR II12011 and II12041. They contain important setup and WebSphere information. To access the APAR information, log on to:

`http://www.as400service.ibm.com/`

Click the **+** (plus sign) next to **Tech Info & Databases->Software Problems - APARS->All Info APARs by Release->Search**. Enter the APAR number, and click **Search**. The search should return the title of the APAR. Click the hyperlink by the title, and read the APAR information.

You should also check the readme file. To access the readme file, log on to:

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

Click **Support** in the left frame. In the right frame, support areas are listed. Click **IBM Net.Commerce** in the **Technical Library** area. Click **OS400** in the header bar. From here, you can view the readme file and the list of PTFs needed for different levels of the OS/400 operating system.

12.1 Creating New Net.Commerce Instances

After installing the Net.Commerce system, you must configure an instance as explained in these steps:

1. Ensure that you are logged on to the AS/400 system using the user ID that was created earlier.
2. To prevent timeouts during long Net.Commerce configuration processes, such as when the database schema is created and populated with sample data, alter the *ADMIN Web server instance as described here:
   a. If the HTTP administration Web server is running, stop the server by using the command:

   ```
   ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
   ```

b. From the command line, type:

```
WRKHTTPCFG *ADMIN
```

c. Add the following line to the end of the configuration file:

```
ScriptTimeOut 120 minutes
OutputTimeOut 120 minutes
```

Note

Make sure both ScriptTimeOut and OutputTimeOut values are set to 120 minutes to ensure a *complete* installation. Otherwise, unpredictable results may occur that make it difficult to debug.

d. Start the HTTP administration Web server. The AS/400 command to invoke this is:

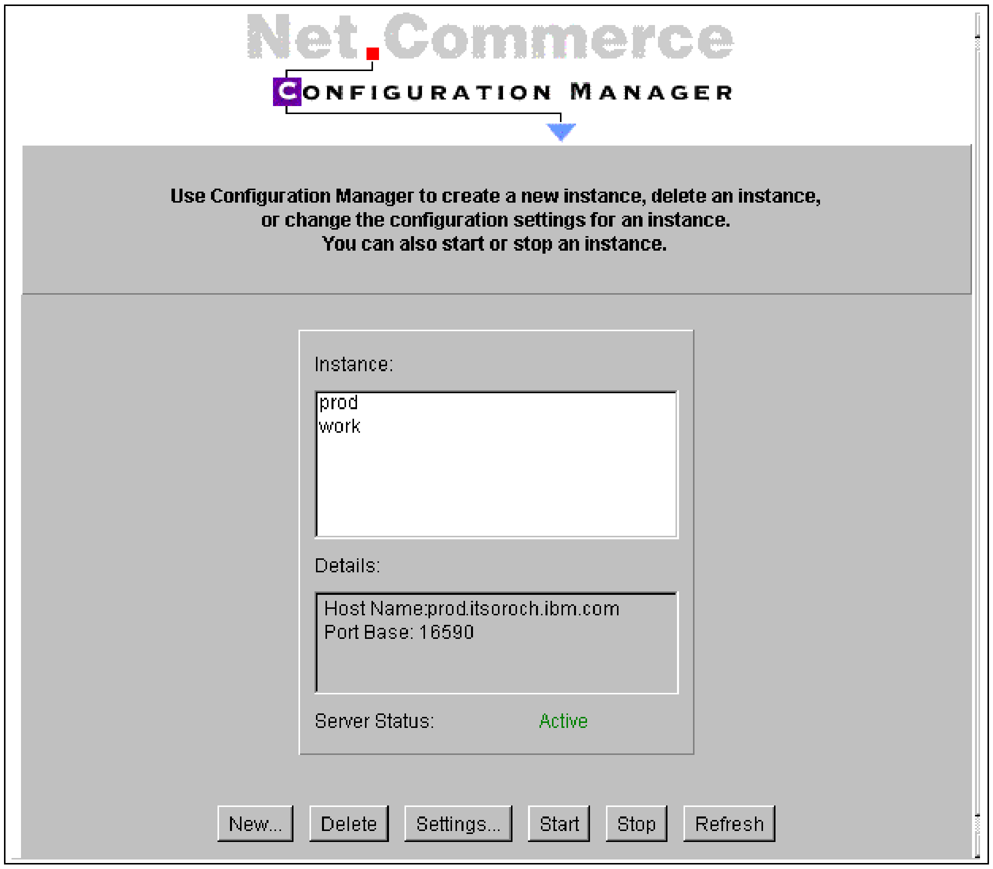
```
STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

3. Start your Web browser, and ensure that the following are disabled:
   - Memory cache
   - Disk cache
   - Proxy servers (also known as socks servers)
4. To display the main Net.Commerce Configuration Manager page, complete the following steps:

a. Type the following URL to display the AS/400 Tasks Page:

```
http://AS01.MYCOMPANY.COM:2001
```

b. Click the **IBM Net.Commerce for AS/400** link. This displays the Configuration Manager page as shown in Figure 121 on page 175.



Use Configuration Manager to create a new instance, delete an instance, or change the configuration settings for an instance. You can also start or stop an instance.

Instance:

prod
work

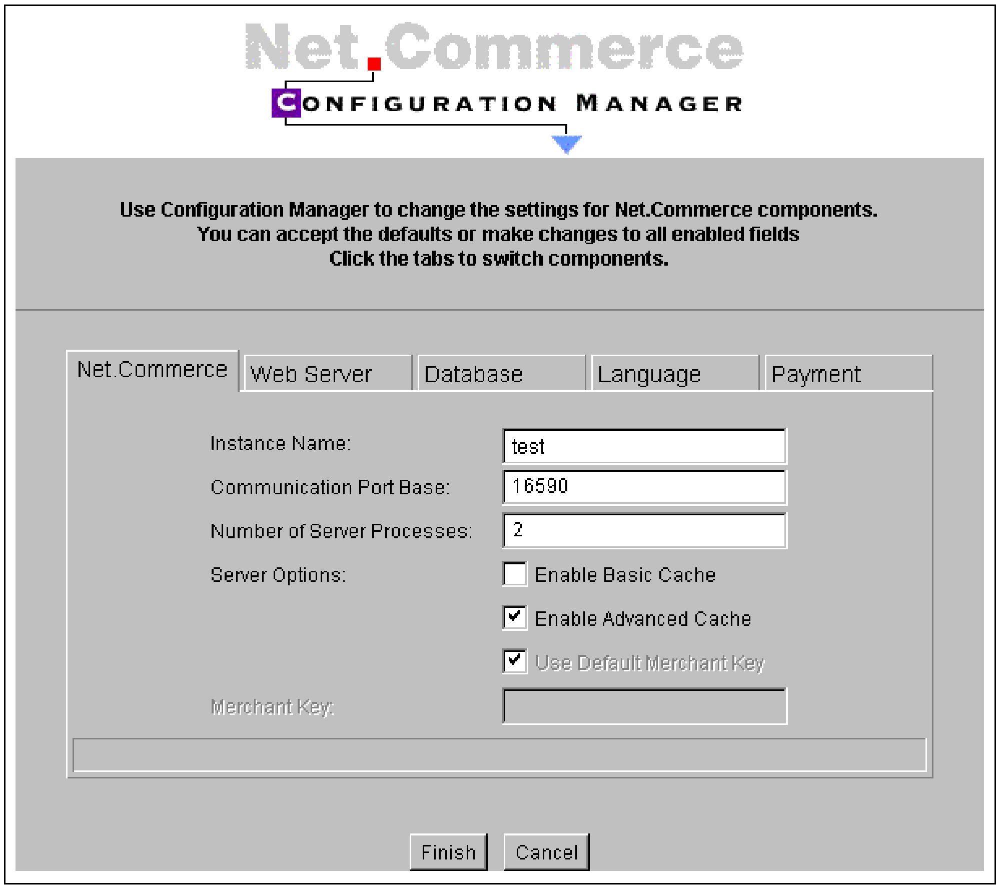
Details:

Host Name:prod.itsoroch.ibm.com
Port Base: 16590

Server Status: Active

Figure 121. Configuration Manager

5. Click on **New** button to display the Settings window.

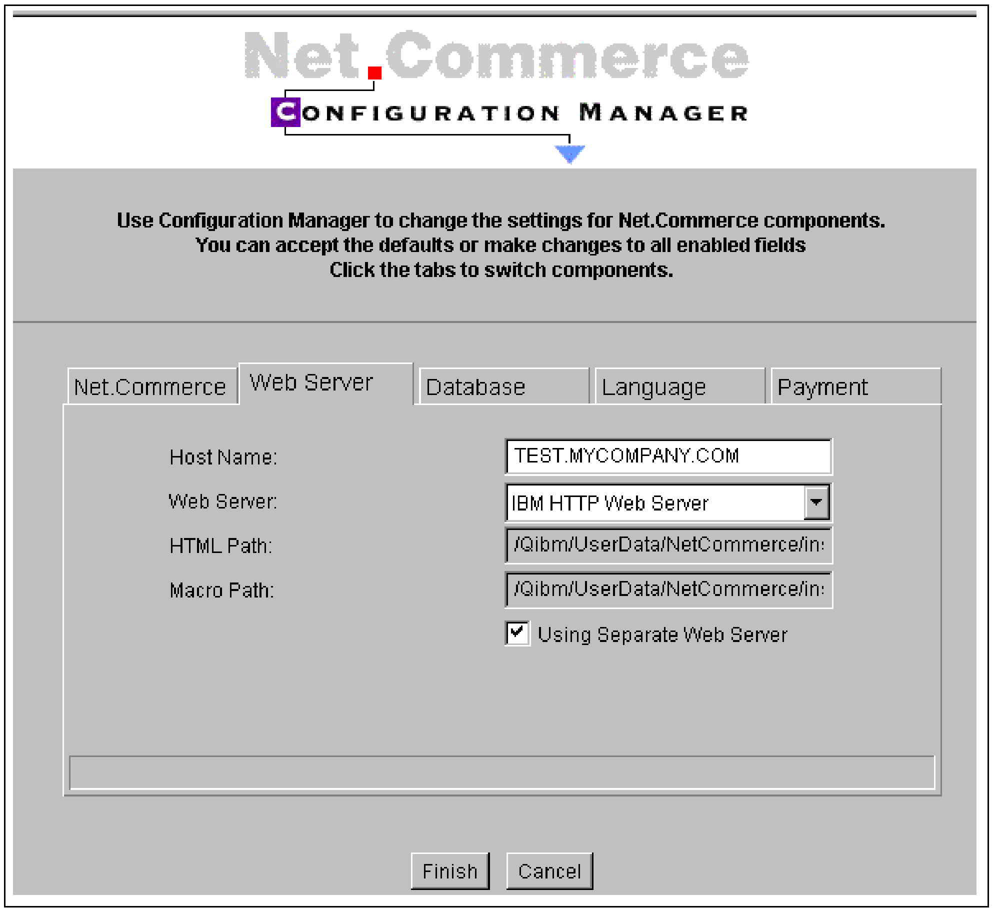


Use Configuration Manager to change the settings for Net.Commerce components.
You can accept the defaults or make changes to all enabled fields
Click the tabs to switch components.

Instance Name: test
Communication Port Base: 16590
Number of Server Processes: 2
Server Options: ☐ Enable Basic Cache
☑ Enable Advanced Cache
☑ Use Default Merchant Key
Merchant Key:

Figure 122. Create a New Net.Commerce Instance

6. To complete the Net.Commerce tab, perform the following steps:
   a. Type `Test` as the Instance Name.
   b. Leave default values for Communication Port Base and Number of Server Processes.
   c. Check the Server Options. For ShopITSO, we checked the **Enable Advanced Cache** and **Use Default Merchant Key** boxes.



Use Configuration Manager to change the settings for Net.Commerce components.
You can accept the defaults or make changes to all enabled fields
Click the tabs to switch components.

Host Name: TEST.MYCOMPANY.COM

Web Server: IBM HTTP Web Server

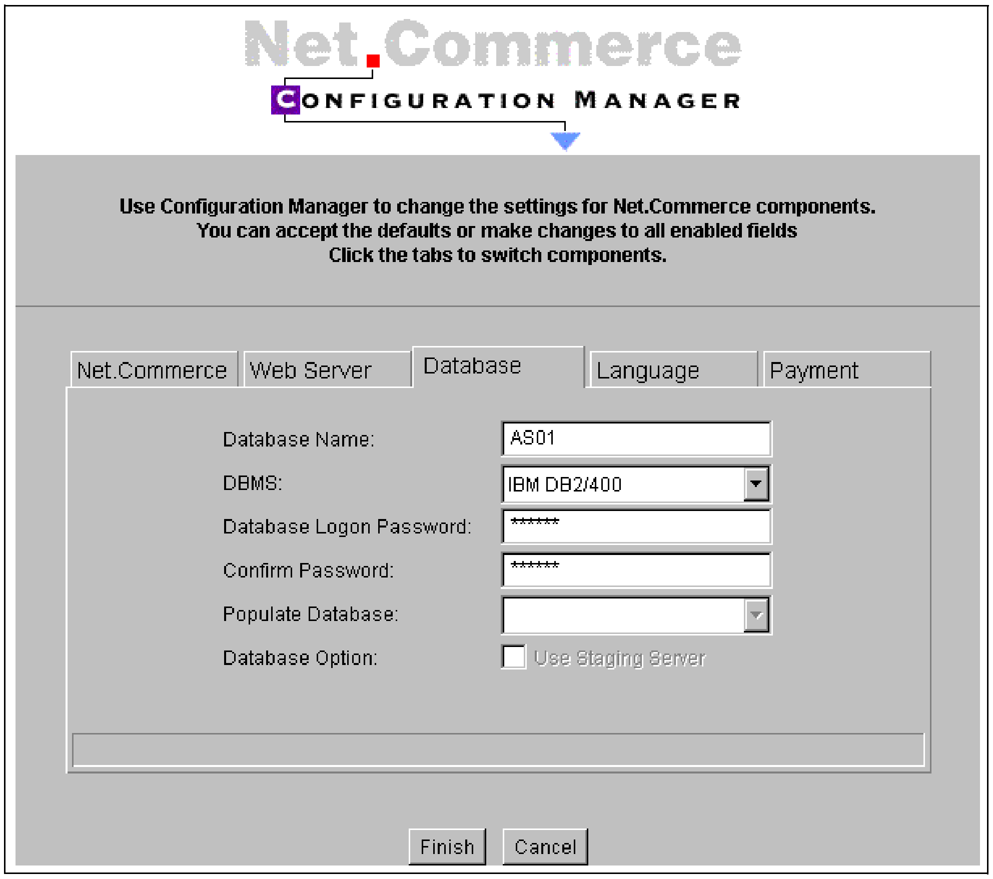
HTML Path: /Qibm/UserData/NetCommerce/in:

Macro Path: /Qibm/UserData/NetCommerce/in:

[✓] Using Separate Web Server

Figure 123. Web Server Tab

7. To complete the Web Server tab, perform the following steps:
   a. Type the fully quantified Web server name: `Test.MYCOMPANY.COM`
   b. Select **IBM HTTP Web Server** as the Web server.
   c. Choose **Using Separate Web Server** to have a unique Web server configuration for each instance.


Net.Commerce CONFIGURATION MANAGER

Use Configuration Manager to change the settings for Net.Commerce components.
You can accept the defaults or make changes to all enabled fields
Click the tabs to switch components.

Database Name: AS01
DBMS: IBM DB2/400
Database Logon Password: ******
Confirm Password: ******
Populate Database:
Database Option: Use Staging Server

Figure 124. Database Tab

8. To complete the Database tab, perform the following steps:

   a. Type `AS01` as the Database Name which can be retrieved from the `WRDBDIRE` command.

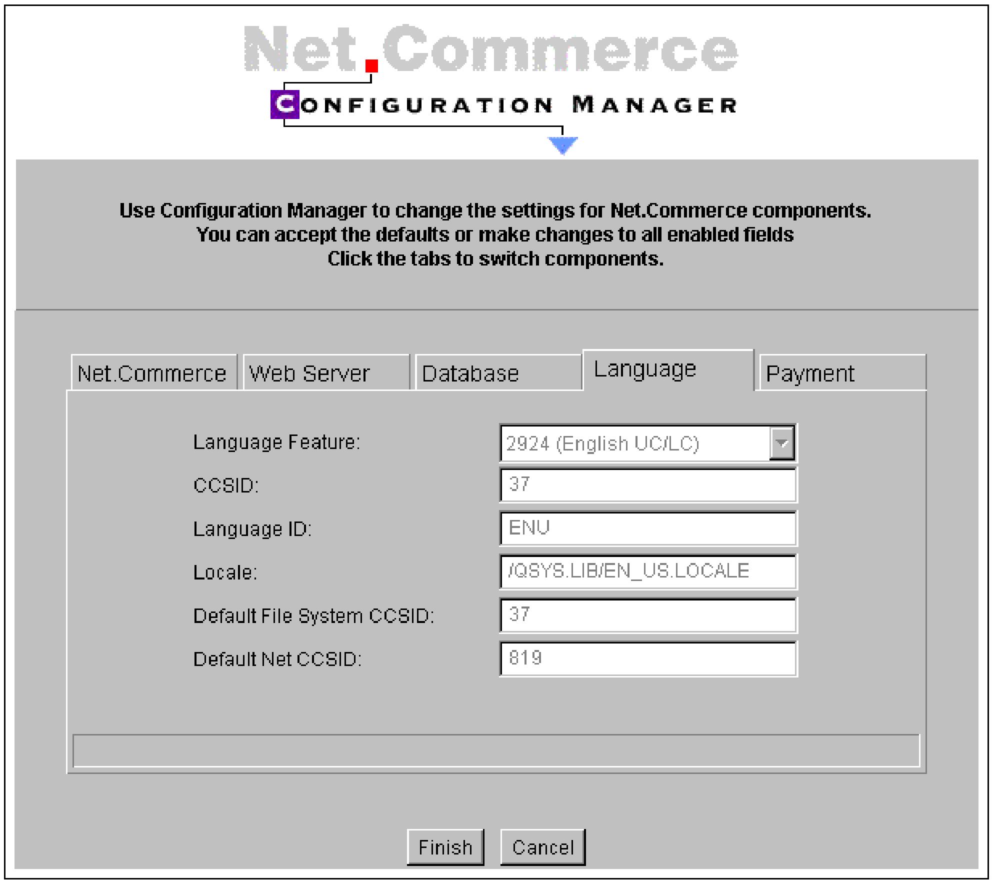
   b. Select **IBM DB2/400** as the DBMS.

   c. Enter a Database Logon Password.

      When you need to change this password, you *must* use the Configuration Manager interface. *Do not* use the AS/400 native interface to change the password. While you can change the password associated with the user profile on the AS/400 system, the Net.Commerce functions that need the password will not be informed of the new password.

   d. Enter the same password in the Confirm Password field.

   e. You may choose to populate the database with sample data.

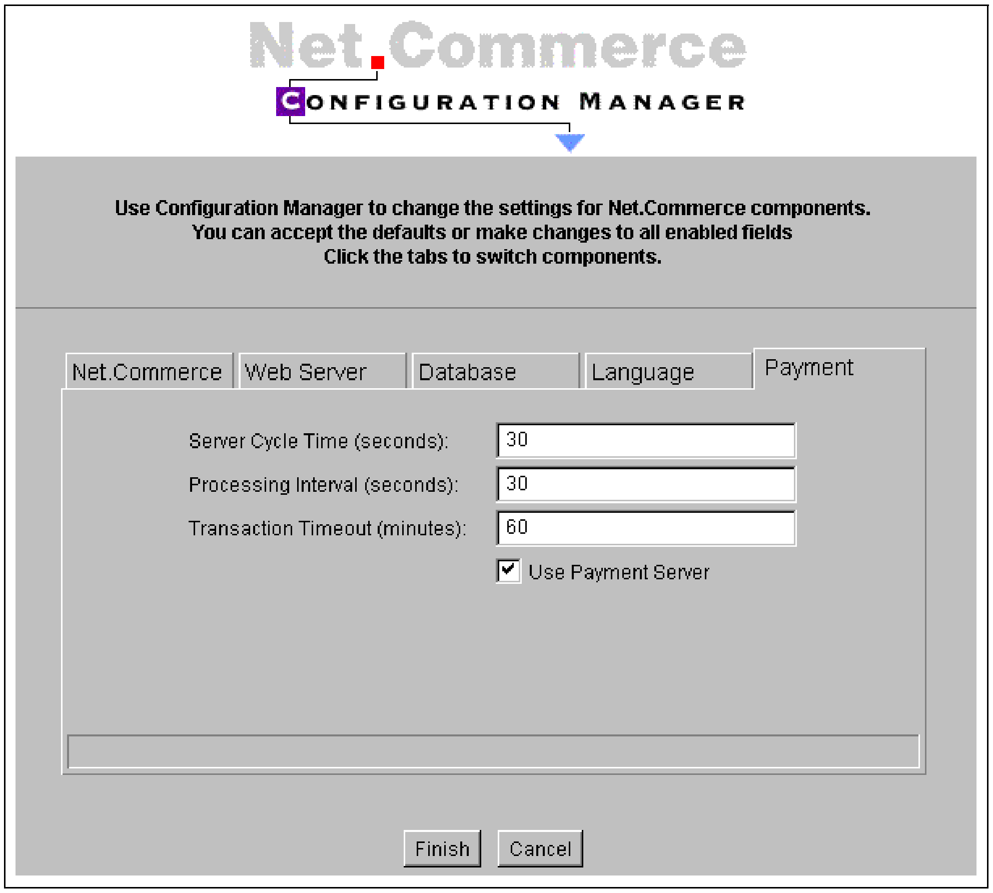


Use Configuration Manager to change the settings for Net.Commerce components.
You can accept the defaults or make changes to all enabled fields
Click the tabs to switch components.

Language Feature:	2924 (English UC/LC)
CCSID:	37
Language ID:	ENU
Locale:	/QSYS.LIB/EN_US.LOCALE
Default File System CCSID:	37
Default Net CCSID:	819

Figure 125. Language Tab

9. Leave the default information for the Language tab.



Use Configuration Manager to change the settings for Net.Commerce components.
You can accept the defaults or make changes to all enabled fields
Click the tabs to switch components.

Server Cycle Time (seconds): 30
Processing Interval (seconds): 30
Transaction Timeout (minutes): 60
[✓] Use Payment Server

Figure 126. Payment Tab

10. To complete the Payment Tab for the Payment Server configuration, perform the following steps:
    a. Only *one* Net.Commerce instance can use Payment Server. Also, the payment server must be installed (`RSTLICPGM 5733-PY1`) and created (`CRTPYMSVR KEYPWD(PASSWORD)`) prior to configuration.
    b. Leave the default settings for Server Cycle Time, Processing Interval, and Transaction Timeout.
11. Click on the **Finish** button.
12. To remove "ScriptTimeOut 120 minutes" and "OutputTimeOut 120 minutes" from the *ADMIN configuration, enter the command:

```
WRKHTTPCFG *ADMIN
```

13. Stop and start the *ADMIN instance with the following commands:

```
ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

14. Use the steps found in 9.5.4, “Configuring Web Server to Use SSL Server Authentication (V4R3)” on page 132, or in 9.5.5, “Configuring Web Server to Use SSL Server Authentication (V4R4)” on page 135, to enable SSL security in the HTTP server for your Net.Commerce instance. *Do not use WRKHTTPCFG in V4R4*. The HTTP configuration name will either be the name you specified when you created the Net.Commerce instance (TEST in

this example), or it will be QNETCOMM. Use the certificate you created in either 9.5, “Creating a Self-signed Certificate” on page 122, or 9.6, “Requesting a Server Certificate from an Internet CA” on page 139.

Note

Before you start your HTTP server, check the *ServerInit* directives in the HTTP configuration. A change may be required depending on the level of WebSphere installed on the system. An incorrect setting results in the termination of the HTTP server jobs for the instance. Refer to the documentation found in Informational APAR II12011 and II12041. For instructions about obtaining these APARs, refer to “Note” on page 173.

15. Start the IBM HTTP instance test using the command:

    ```
    STRTCPSVR SERVER(*HTTP) HTTPSVR(TEST)
    ```

    Or, start the instance from the AS/400 Task Page.

16. Start the Net.Commerce instance with the following command:

    ```
    STRNETCSVR INSTANCE(TEST)
    ```

    The HTML path populates automatically by the command or from the AS/400 Task Page.

To set up the instances "Work" and "Prod", follow steps 1 through 16 and substitute "test" with the appropriate instance name. Please be sure to have a unique Web server instance with a unique IP address for each Net.Commerce instance. You should skip the steps to remove the ScriptTimeOut and OutputTimeOut directives from the *ADMIN server (steps 12 and 13) until after you build all of your instances.

12.2 Deleting Net.Commerce Instances

To delete a Net.Commerce instance, you must use the following steps:

1. Stop the Net.Commerce instance, if it is running, with the command:

   ```
   ENDNETCSVR <(Instance Name)>
   ```

2. Stop the Web server if it is running with the command:

   ```
   ENDTCPSVR SERVER(*HTTP) HTTPSVR(<Instance Name>)
   ```

3. Delete the Net.Commerce instance using the Configuration Manager accessed from the AS/400 Task Page.
4. Delete the instance library named <instance_name> using the command:

   ```
   DLTLIB <instance_name>
   ```

5. Delete the user profile that was created during the configuration of the deleted Net.Commerce instance with the command:

   ```
   DLTUSRPRF <Instance_Name> OWNOBJOPT(*DLT)
   ```

   This deletes the user profile and all of the objects that the user profile owns, including items in the root file system. Several attempts may be required to delete all of the owned objects.

6. Delete the directories and files that are associated with the deleted instance within the IFS directory: \QIBM\USERDATA\NETCOMMERCE\INSTANCE\<instance_name>

12.3 Deleting Net.Commerce Licensed Program Product

If you want to delete the licensed program product (LPP), you must end all jobs that have a lock on the QNETCOMM library. To detect jobs that are locking the library, issue the command:

```
WRKOBJLCK OBJ(QNETCOMM) OBJTYPE(*LIB)
```

End the QNETCOMM subsystem with the command:

```
ENDSBS(QNETCOMM)
```

To delete the program product, issue the command:

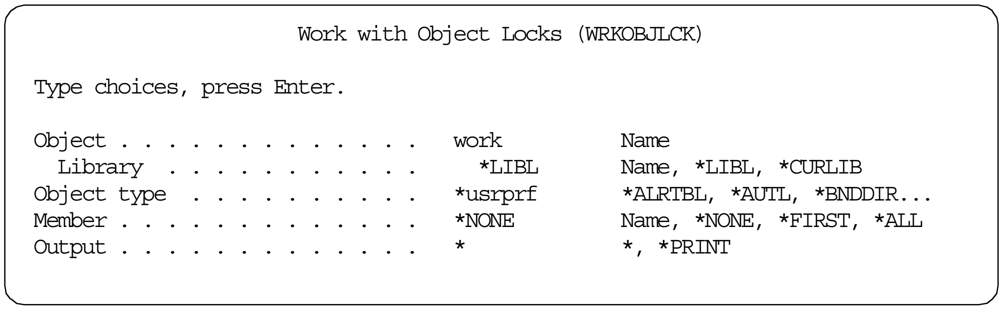
```
DLTLICPGM 5798NC3
```

12.4 Database Server Problem Determination Procedure

Some of the Net.Commerce features, such as product advisor, use JDBC to connect to the Net.Commerce database schema. The database server job's job log can assist you in problem determination of ODBC and JDBC connection.

Use the following procedure to locate the relevant database server job and display its job log:

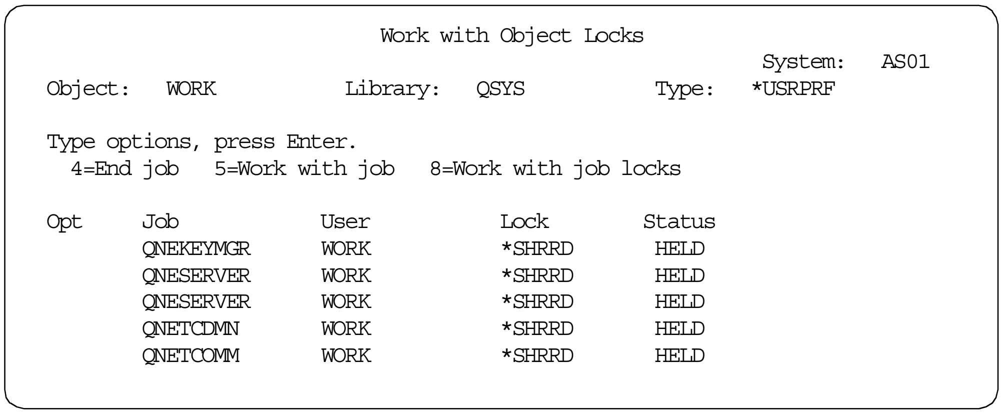
1. Type `WRKOBJLCK` and press **F4.** Fill the parameters as shown in Figure 127. Replace the Object keyword value (work) with your instance user profile.



```
                     Work with Object Locks (WRKOBJLCK)

 Type choices, press Enter.

 Object . . . . . . . . . . . . .   work         Name
   Library  . . . . . . . . . . .     *LIBL      Name, *LIBL, *CURLIB
 Object type  . . . . . . . . . .   *usrprf      *ALRTBL, *AUTL, *BNDDIR...
 Member . . . . . . . . . . . . .   *NONE        Name, *NONE, *FIRST, *ALL
 Output . . . . . . . . . . . . .   *            *, *PRINT
```

Figure 127. Find the Serving Database Job

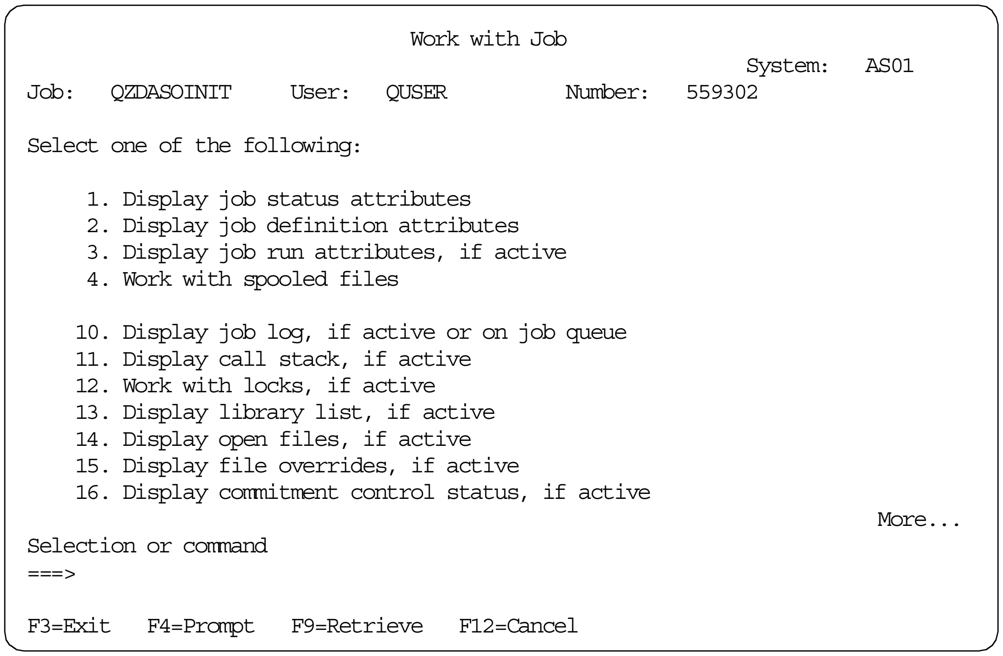
2. Press the **Enter** key. You will see a list of jobs on the screen as shown in Figure 128.



```
                              Work with Object Locks
                                                                System:   AS01
 Object:   WORK          Library:   QSYS          Type:   *USRPRF

 Type options, press Enter.
   4=End job   5=Work with job   8=Work with job locks

 Opt     Job            User           Lock       Status
         QNEKEYMGR      WORK           *SHRRD      HELD
         QNESERVER      WORK           *SHRRD      HELD
         QNESERVER      WORK           *SHRRD      HELD
         QNETCDMN       WORK           *SHRRD      HELD
         QNETCOMM       WORK           *SHRRD      HELD
```

Figure 128. List of Jobs Used by Net.Commerce Instance User

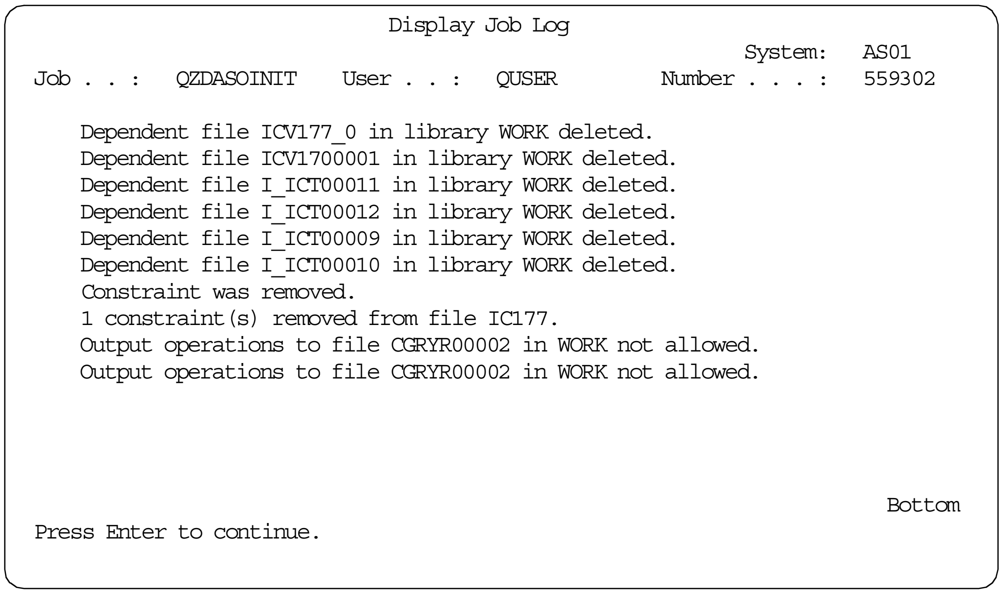
3. The jobs named QZDASOINIT are the database server jobs. Type `5` to the left of this job and press **Enter**. You will see the work with jobs menu as shown in Figure 129 on page 183.



```
                               Work with Job
                                                            System:   AS01
Job:   QZDASOINIT     User:   QUSER         Number:   559302

Select one of the following:

     1. Display job status attributes
     2. Display job definition attributes
     3. Display job run attributes, if active
     4. Work with spooled files

    10. Display job log, if active or on job queue
    11. Display call stack, if active
    12. Work with locks, if active
    13. Display library list, if active
    14. Display open files, if active
    15. Display file overrides, if active
    16. Display commitment control status, if active
                                                                   More...
Selection or command
===>

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Figure 129. Work with the Database Server

4. Choose option **10**, Display job log. Press **Enter** to see the database server job log as shown in Figure 130 on page 184.



```
                                Display Job Log
                                                              System:   AS01
 Job . . :   QZDASOINIT    User . . :   QUSER        Number . . . :   559302

     Dependent file ICV177_0 in library WORK deleted.
     Dependent file ICV1700001 in library WORK deleted.
     Dependent file I_ICT00011 in library WORK deleted.
     Dependent file I_ICT00012 in library WORK deleted.
     Dependent file I_ICT00009 in library WORK deleted.
     Dependent file I_ICT00010 in library WORK deleted.
     Constraint was removed.
     1 constraint(s) removed from file IC177.
     Output operations to file CGRYR00002 in WORK not allowed.
     Output operations to file CGRYR00002 in WORK not allowed.




                                                                          Bottom
 Press Enter to continue.
```

Figure 130. Database Server Job Log

In the job log screen, you can see a detailed message by pressing **F10.**

The work with job for the database server allows you to perform more operations such as changing the priority or start debug by using the `STRSRVJOB` command.

Chapter 13. Building the Mall and Store

There are two different types of Net.Commerce site that can be created:

- A *multiple store* site or mall contains a number of stores that shoppers access from a single mall home page or directory. In Net.Commerce, it is a single Net.Commerce instance with multiple stores configured in the same database. In this system, there will be multiple merchants and administrators. The shopping process may be customized for each merchant.
- A *single store* site is a store that exists independently of any other online store or mall. In Net.Commerce, it is a single Net.Commerce instance with only one store configured in the database. In this system, there will be one merchant, one shopping process, and perhaps only one administrator.

The creation of the stores within the site can be done in two different ways for either type of site, either using Store Creator or using functions of Site Manager and Store Manager. See also Chapter 21, "Site Administration" on page 453.

13.1 Net.Commerce Sample Stores

Net.Commerce provides samples that you can use as starting points for creating stores or malls. Each sample shows how you can use specific features of Net.Commerce to build a customized store or mall. Each is intended to help you generate business ideas.

Also the East West Food Mart tutorial is delivered with the Net.Commerce product. This tutorial is intended for the administrator who will build and manage a Net.Commerce store. It provides step-by-step instructions on how to create a store from a DB2 database, by using the Site Manager and the Store Manager in the Administrator interface, the mass import utility, and by writing and customizing your own Net.Data macros. It also describes how to add unique store features using Java Script. You will find this tutorial in the directory `/QIBM/ProdData/NetCommerce/html/MRIxxxx/ncbooks`, where xxxx stands for your language code (for example, 2924 for the English language).

The following Net.Commerce samples are available:

- **Metropolitan mall (demomall)** — Demonstrates a mall scenario that implements the basic features of Net.Commerce with minimal customizing.
- **Euro mall** — Is similar to the Metropolitan mall. It uses the same products and has the same stores as the Metropolitan mall. It also demonstrates the use of the euro currency.
- **East West Food Mart** — Demonstrates a business-consumer scenario that has been customized to create a simple and expedient shopping flow.

To view a sample, you first need to select the sample database under the Database tab, during instance configuration. Then, you can view it through your browser by typing in its URL. See Figure 124 on page 178 in 12.1, "Creating New Net.Commerce Instances" on page 173.

The Euro mall is intended to demonstrate some of the features and functions that you can use to include euro support in your own mall. The Euro mall comes complete with sample data and hypothetical conversion rates. You can view and try out the Euro mall to see how an euro-enabled mall can function, and modify your own mall based on the Euro mall.

The mall contains the following stores that you can replace or modify:

- **6th Avenue** — A department store that sells hardware, computers, and clothing. All product prices are stored in the database in EUR. This store accepts several shopping currencies: ATS, BEF, DEM, IEP, ITL, LUF, PTE, ESP, FRF, and EUR. All shopping currencies are dual displayed using EUR as the counter value currency. For FRF, in addition to EUR, this store displays FIM, ITL, and NLG as counter value currencies.
- **Basics** — A clothing store for men which specializes in pants and tops. This store accepts EUR and FRF as the shopping currencies. In addition, by overriding the mall-level currency formatting information in the table CURRFORMAT with merchant-specific values, this store displays the euro symbol as a GIF file, rather than using the mall level default of the euro. The euro symbol works only when shoppers use browsers that support HTML 4.0 and operating systems that can display the euro symbol.
- **Next Generation** — A computer store that features Product Advisor for helping the shopper find computer equipment to purchase. All product prices are in FRF. Therefore, the store accepts FRF as the shopping currency. This store also presents monetary amounts through dual display of EUR and FRF.
- All other stores are under construction and do not contain data. These stores are provided as placeholders.

13.2 Store Creation Choices

As mentioned before, you have two choices to create a store. You can use the Store Creator or the Site and Store Manager functions. Of course, using the Store Creator is the easy way to get a new store.

Figure 131 on page 187 shows the differences. With Store Creator, you have only to complete the processes for product catalog creation, assign Shopper Groups (your shop has no Shopper Groups), define Discounts (in our shop, the pricing function comes from the back-end system), and define SET Payment.

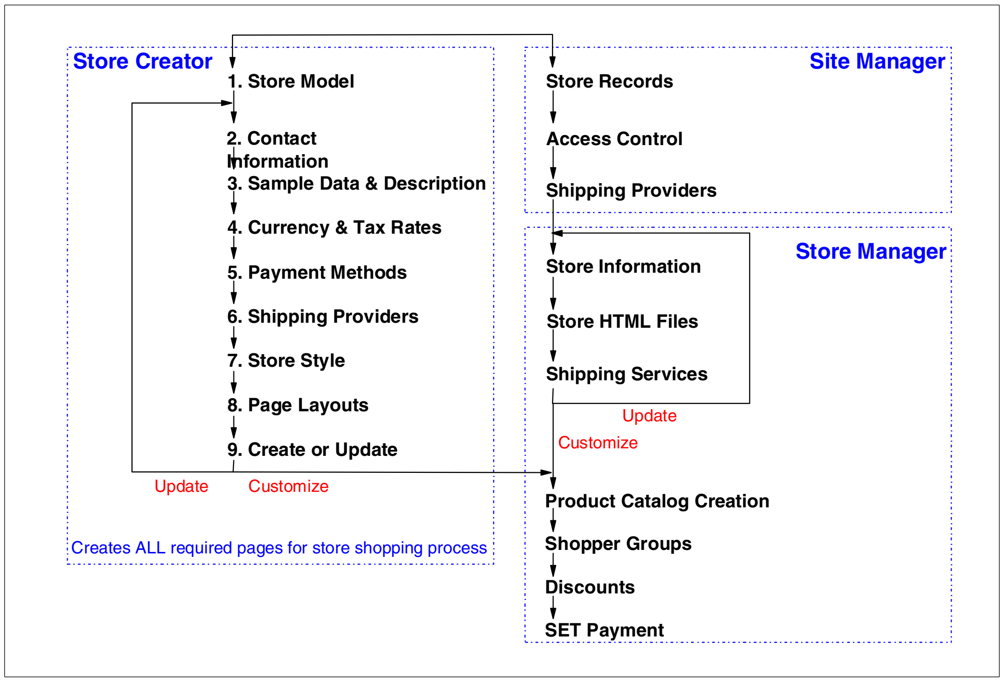


Figure 131. Store Creation Choices

13.3 Building the Store with Store Creator

Store Creator provides an easy-to-use graphical application that steps a user through the creation of a store in nine easy steps. It supports three store models:

- **One Stop Shop** — A simple business-to-consumer store model that does not require shoppers to register.
- **Personal Delivery** — A business-to-consumer store model that offers more shopper functions including registration and address book.
- **Business to Business** — A business-to-business store model that includes an approvals process for purchases.

Store Creator is only accessible by an administrator in the Site Administrators or Store Creators access group.

Figure 132 on page 188 through Figure 134 on page 189 show the three different store models One Stop Shop, Personal Delivery, and Business to Business.

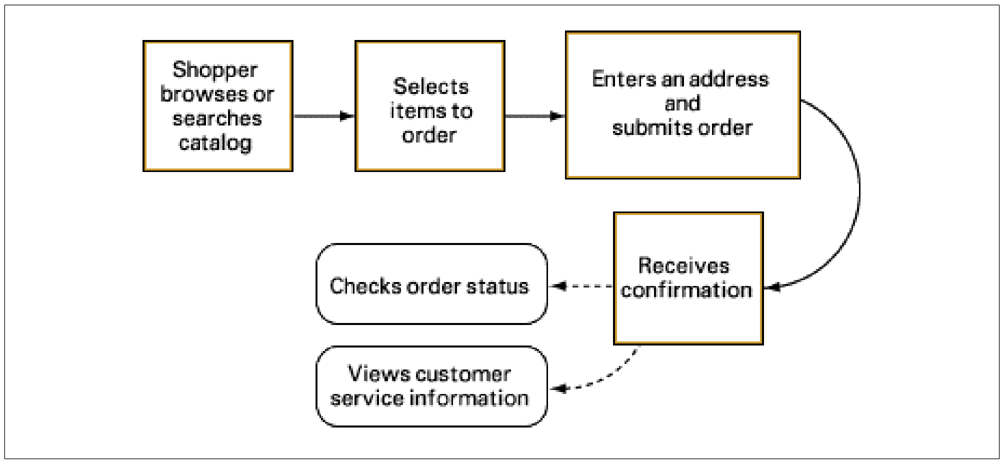

Shopper browses or searches catalog
Selects items to order
Enters an address and submits order
Receives confirmation
Checks order status
Views customer service information

Figure 132. One Stop Shop Store Model

The One Stop Shop model works without shopper registration. The customer has no address book, and the ship-to and bill-to addresses are the same for the whole order.

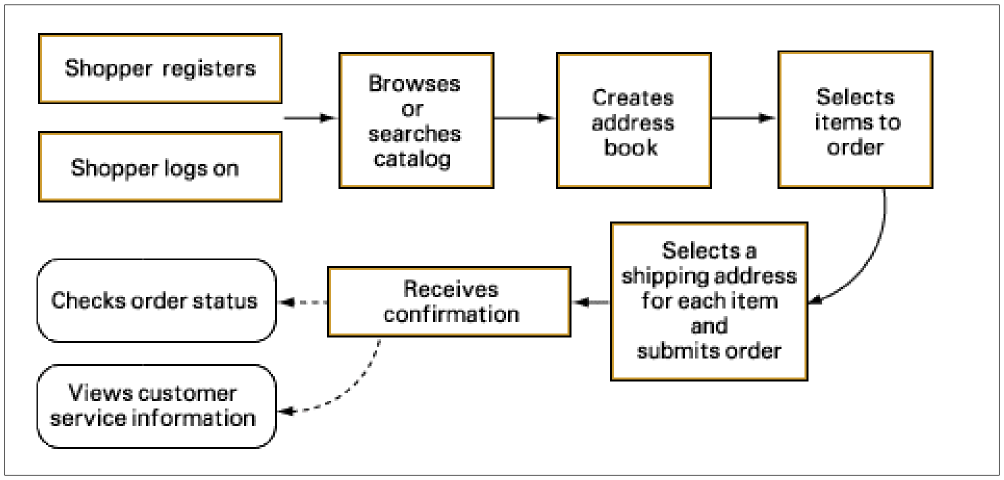

Shopper registers
Shopper logs on
Browses or searches catalog
Creates address book
Selects items to order
Selects a shipping address for each item and submits order
Receives confirmation
Checks order status
Views customer service information

Figure 133. Personal Delivery Store Model

The Personal Delivery model works with shopper registration. The customer can work with an address book. Shoppers place items in an order list. These items are kept there until the shopper orders them or removes them. If the shopper has done neither, the items remain in the order list for future visits.

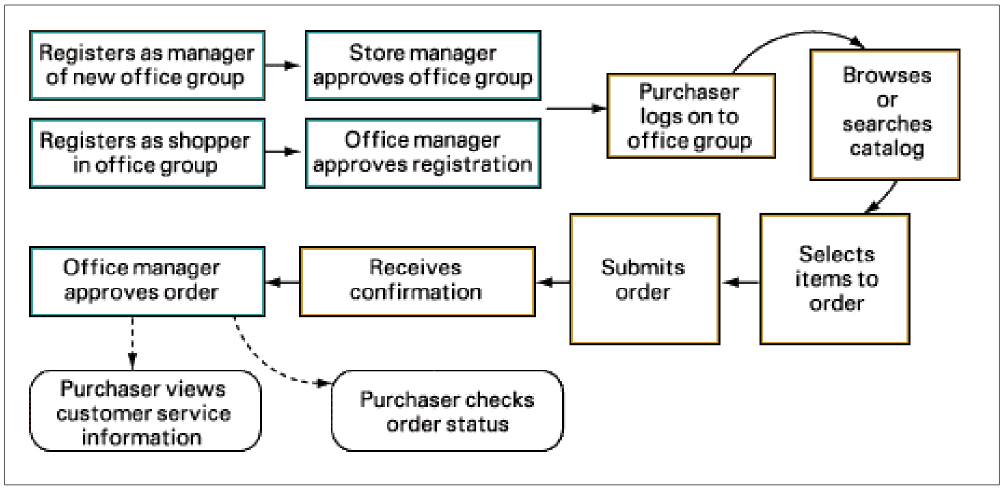

Registers as manager of new office group
Store manager approves office group
Registers as shopper in office group
Office manager approves registration
Purchaser logs on to office group
Browses or searches catalog
Selects items to order
Submits order
Receives confirmation
Office manager approves order
Purchaser views customer service information
Purchaser checks order status

Figure 134. Business to Business Shop Model

The Business to Business store model has a corporate purchasing structure. It allows merchants to group their business clients into different shopper groups, called *office groups*, which can be used to target the groups with different product offerings and prices.

The store manager is responsible for approving office groups and defining the product prices and product pages that will be available to each group. Before making a purchase, each shopper must register with an office group and receive approval from the office group manager.

13.3.1 Objects Built by Store Creator

The Net.Data macros and HTML source files, which are built when you use the Store Creator function, are located in the following IFS (Integrated File System) directories (path):

- **HTML source file:**

  /QIBM/UserData/NetCommerce/instance/*inst_name*/html/index.html

- **Macros for category displays**:

  /QIBM/UserData/NetCommerce/instance/*inst_name*/macro/category/*Shop_name*/cat1.d2w

- **Macros for product display**:

  /QIBM/UserData/NetCommerce/instance/*inst_name*/macro/product/*Shop_name*/prod1.d2w

- **All other macros for controlling your e-business and the include file**:

  /QIBM/UserData/NetCommerce/instance/*inst_name*/macro/*Shop_name*/

Note

Store Creator also creates a include file with the name of *shop_name*.inc. This include file delivers define statements that are used in all Net.Data macros for this shop. For example, there is a define value for the merchant number. Therefore, it is not necessary to retrieve this number from the DB2/400 database.

13.4 Building the Store with Site and Store Management Functions

If Store Creator is *not* used to create stores, then the Site Administrator uses the following Site Manager functions.

For a *Mall*, use the following Site Manager functions:

- **Mall Information form** — Used to enter information about the mall, and create a directory of stores.
- **Store Records form** — Used to create the store record in the Net.Commerce database. In a Mall, this is used multiple times.
- **Shipping Providers form** — Compiles a list of shipping providers. In a Mall, stores selects which shipping providers they will support from this list.

Other *Site Manager forms* that may be required include:

- **Access Control form** — Used to create administrator users and assign them to an Access Group for a particular store or site administration or store creation.
- **Shopper Information form** — Used to create or modify shoppers. For example, a shopper may have forgotten their password.
- **Currency Mapping form** — Used to create a list of currencies for the SET payment support.
- **Task Management form** — Used to override functions and macros for the shopping process tasks with a customized version for a store.
- **Access Groups form** — Used to create new access groups that can perform specific Net.Commerce commands.
- **Command Security form** — Used to set whether SSL security or logon authentication is required to run particular commands.

Site Manager is only accessible by an administrator in the Site Administrators access group.

Store Manager allows you to create and manage all aspects of the store specific information. It is used to perform the following tasks:

- Enter or change store and merchant information
- Create and manage product categories
- Create and manage product information
- Define discounts
- Manage shipping services
- Create and manage shopper groups
- View shopper information and create customer information

- Manage orders
- Configure and manage payments

Store Manager is only accessible by an administrator in the Site Administrators or Store Administrators access group. The payment functions are also accessible by an administrator in the Payment Administrators access group. A Store Administrators access group exists for each store so that each administrator can be restricted to one or more store's data.

Template Designer is a Java applet for designing mall and store Web pages. It can be used to build HTML pages (for example Home pages), category, and product pages.

For more information about Template Designer, see 14.4, "Using Template Designer to Customize Product Advisor Pages" on page 299.

13.5 Implementing the ShopITSO Sample Solution

After the first steps described in Chapter 11, "Installing Net.Commerce" on page 171, and Chapter 12, "Configuring Net.Commerce" on page 173, we are ready to create our sample store. To implement ShopITSO, described in 3.9, "Design of the ShopITSO Sample Solution" on page 57, we use the Store Creator and create our shop with the One Stop Shop store model.

The One Stop Shop store model, as shown in Figure 135, meets the navigation flow in the shop according to our design objectives best. To implement our shop as we designed it, we had to modify the Net.Data macros generated by Store Creator, for example, to implement the possibility to enter quantity for the product to buy.

Another aspect to work with the One Stop Shop is the way the shop pages are presented. One Stop Shop uses frames. This also is a requirement in our design.

The fact that Store Creator creates an include file (see "Note" on page 190) was also a motive to use the Store Creator with the One Stop Shop model.

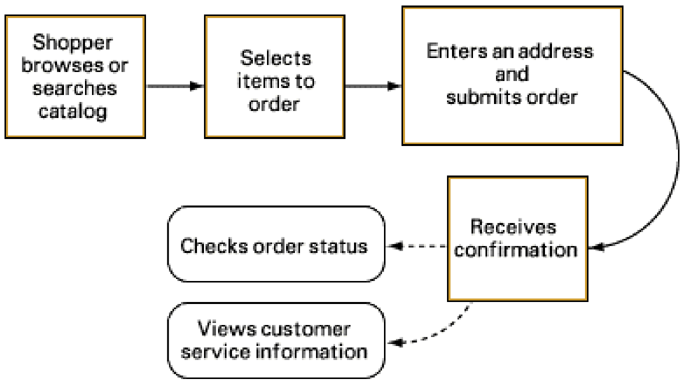

Shopper browses or searches catalog
Selects items to order
Enters an address and submits order
Receives confirmation
Checks order status
Views customer service information

Figure 135. One Stop Shop Store Model

To become familiar with the Net.Data macros, we also looked at the macros that are generated by the Net.Commerce demos, such as the Metropolitan mall and macros that were generated by the Store Creator using the Personal Delivery store model. For more information about the sample stores, see 13.1, “Net.Commerce Sample Stores” on page 185, and the online help from Net.commerce.

Note

Creating several stores with different models through the Store Creator is possible. Doing this in the implementation phase is a good approach. It allows you to set up as many stores as you want for testing purposes, and delivers you a pro store model different Net.Data macros. All stores work with the same system name in the URL. Only the store name that follows the system name in the URL are different.

The following sections describe the steps that we completed to build our sample ShopITSO in detail.

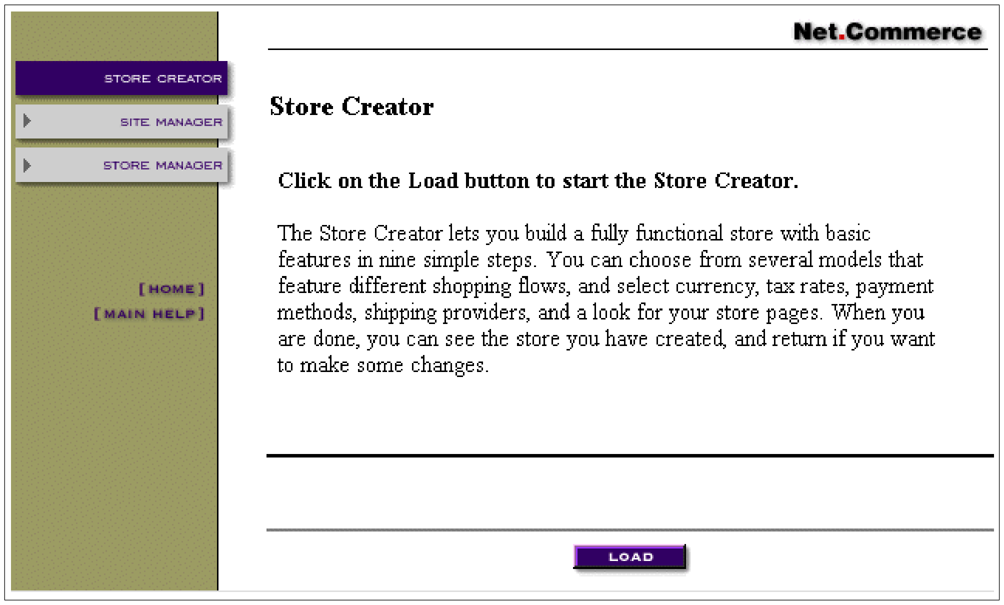
13.6 Building the ShopITSO Sample Store with Store Creator

To create our sample store, we followed this sequence of events:

1. Login as `ncadmin` in the Net.Commerce Administrator Web page at the following URL:

   http://*fullyqualifiedhost_name*/ncadmin/

2. Click on **STORE CREATOR**.
3. Load the Store Creator applet from the Store Creator view, shown in Figure 136, by clicking in the **Load** button. Two windows pop up: the Store Creator Status window and the Store Creator window.



Store Creator

Click on the Load button to start the Store Creator.

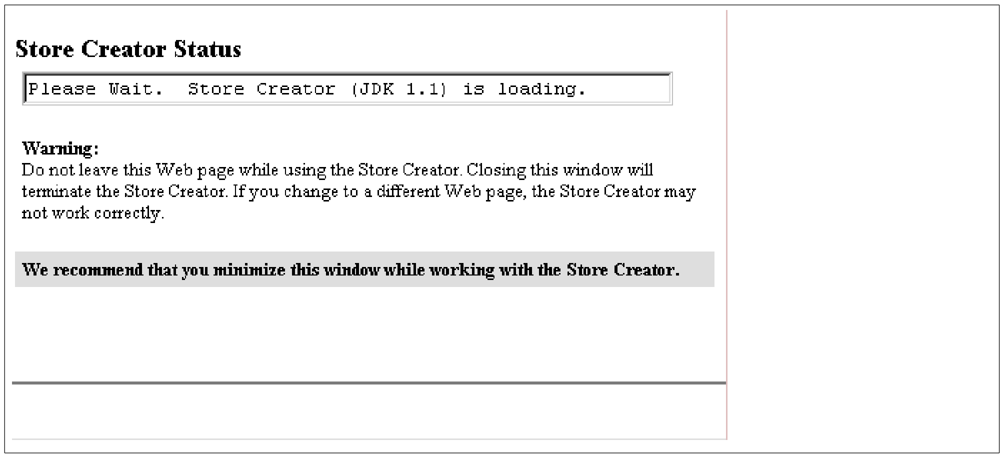
The Store Creator lets you build a fully functional store with basic features in nine simple steps. You can choose from several models that feature different shopping flows, and select currency, tax rates, payment methods, shipping providers, and a look for your store pages. When you are done, you can see the store you have created, and return if you want to make some changes.

Figure 136. Store Creator View in the Net.Commerce Administrator Page

4. Minimize the Store Creator Status window, shown in Figure 137 on page 193, while working with the Store Creator window.

Important

Do not close the Store Creator Status window. If you close this window, the Store Creator window will also close!



Store Creator Status

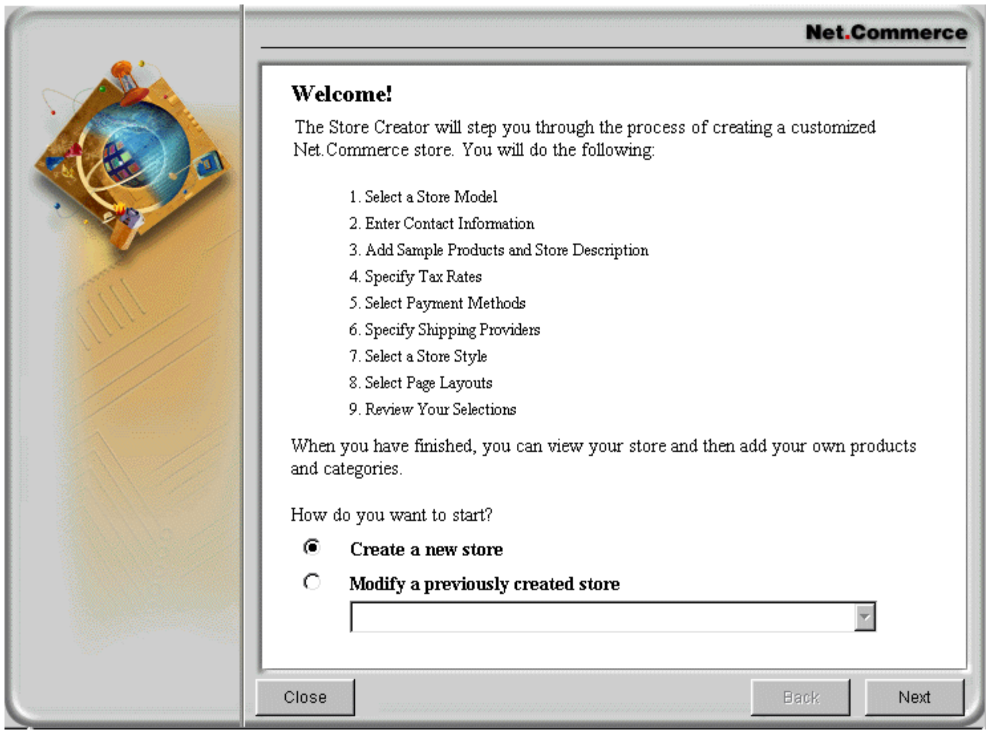
```
Please Wait.  Store Creator (JDK 1.1) is loading.
```

Warning:
Do not leave this Web page while using the Store Creator. Closing this window will terminate the Store Creator. If you change to a different Web page, the Store Creator may not work correctly.

We recommend that you minimize this window while working with the Store Creator.

Figure 137. Store Creator Status Window

5. From the Store Creator window, shown in Figure 138, select **Create a new store** and click the **Next** button.



Welcome!

The Store Creator will step you through the process of creating a customized Net.Commerce store. You will do the following:

1. Select a Store Model
2. Enter Contact Information
3. Add Sample Products and Store Description
4. Specify Tax Rates
5. Select Payment Methods
6. Specify Shipping Providers
7. Select a Store Style
8. Select Page Layouts
9. Review Your Selections

When you have finished, you can view your store and then add your own products and categories.

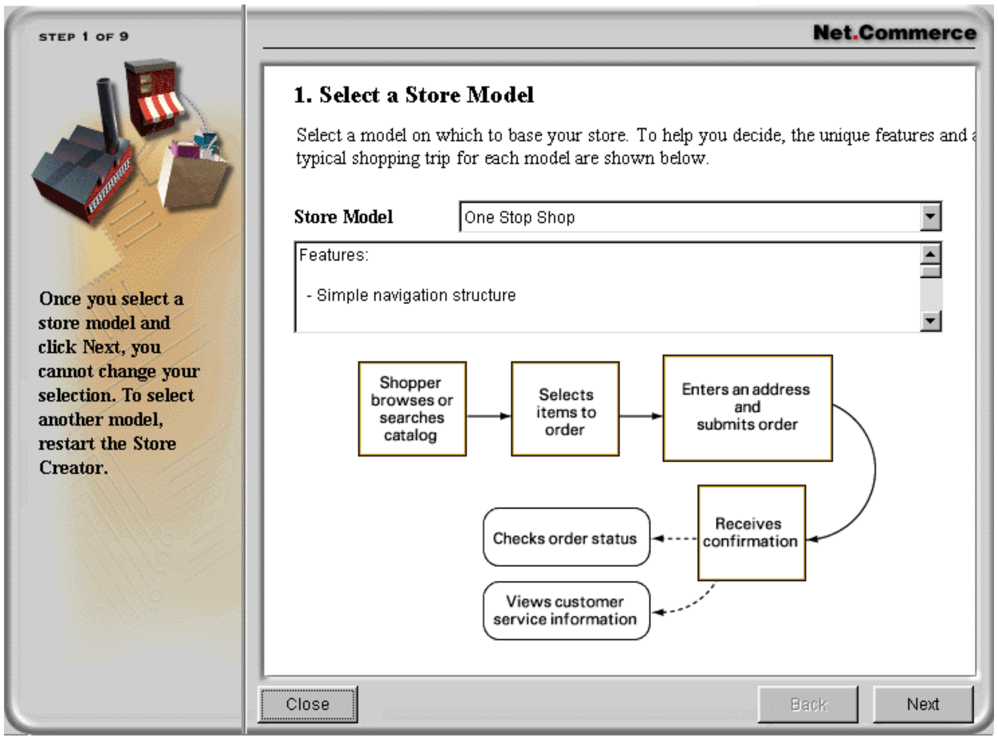
How do you want to start?

● **Create a new store**

○ **Modify a previously created store**

Figure 138. Store Creator — Welcome Window

6. Select **One Stop Shop** for the store model, as shown in Figure 139 on page 194. Click the **Next** button.


STEP 1 OF 9
Net.Commerce
Close
Back
Next

1. Select a Store Model

Select a model on which to base your store. To help you decide, the unique features and a typical shopping trip for each model are shown below.

Store Model: One Stop Shop

Features:

- Simple navigation structure

Shopper browses or searches catalog
Selects items to order
Enters an address and submits order
Receives confirmation
Checks order status
Views customer service information

Once you select a store model and click Next, you cannot change your selection. To select another model, restart the Store Creator.

Figure 139. Store Creator — Step 1. Select a Store Model

7. Enter the contact information, as shown in Figure 140 on page 195, and click the **Next** button.

Important

The value you enter in the Store Name is used as part of the URL name and also the directory names in the IFS, where all the macros are stored. It is the name that follows the system name in a URL.

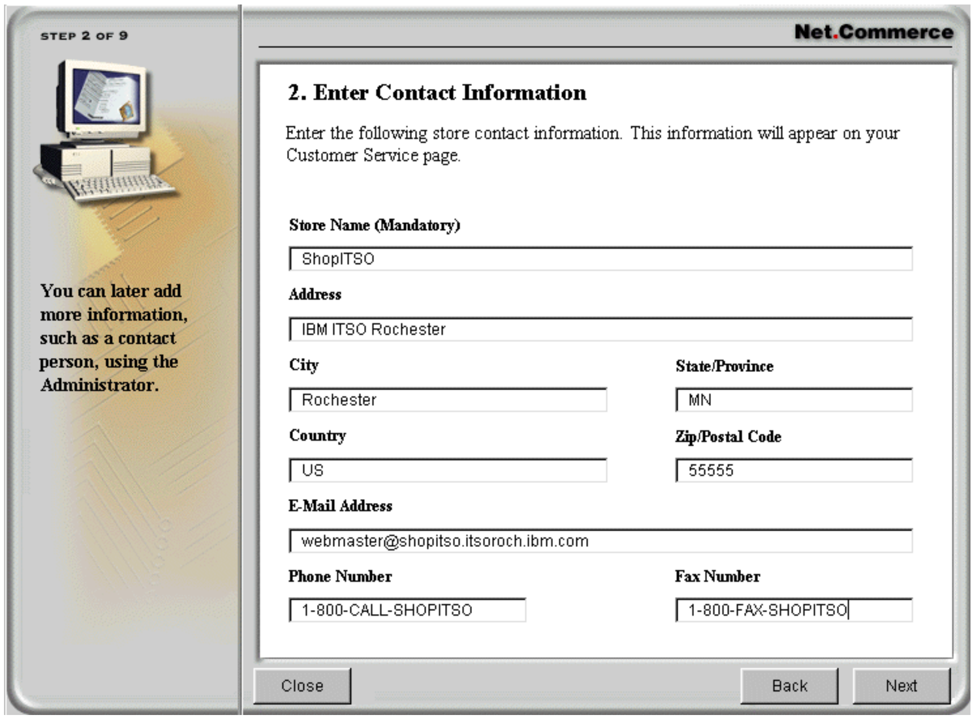
For example, `http://myServer/ShopITSO where ShopITSO` is the name that we entered in the Store Name field. Any spaces you typed in the Store Name are dropped.

When you use a mix of small and capital letters, be sure that the Pass directive in the HTTP configuration matches this. URLs are always case sensitive.

For our example above, we added the following Pass directive in the HTTP configuration file:

```
Pass /shopitso/* /QIBM/UserData/NetCommerce/instance/instance_name/html/ShopITSO/*
Pass /shopITSO/* /QIBM/UserData/NetCommerce/instance/instance_name/html/ShopITSO/*
Pass /SHOPITSO/* /QIBM/UserData/NetCommerce/instance/instance_name/html/ShopITSO/*
```

With these three Pass directives, the customer can work with all three variations of the URLs: `http://myServer/shopitso`, `http://myServer/shopITSO`, and `http://myServer/SHOPITSO`.



2. Enter Contact Information

Enter the following store contact information. This information will appear on your Customer Service page.

You can later add more information, such as a contact person, using the Administrator.

Store Name (Mandatory)	ShopITSO
Address	IBM ITSO Rochester
City	Rochester
State/Province	MN
Country	US
Zip/Postal Code	55555
E-Mail Address	webmaster@shopitso.itsoroch.ibm.com
Phone Number	1-800-CALL-SHOPITSO
Fax Number	1-800-FAX-SHOPITSO

Figure 140. Store Creator — Step 2. Enter Contact Information

8. Select **No Sample Products** in the Sample Products selection list to create a store with no generated sample products. Enter the store front description (using HTML tags if needed), as shown in Figure 141 on page 196, and click the **Next** button.

   There are four bundles of sample data provided as mass import files that the store creator will automatically import. The *No Sample Product* bundle just creates a root category named *Top Category*.

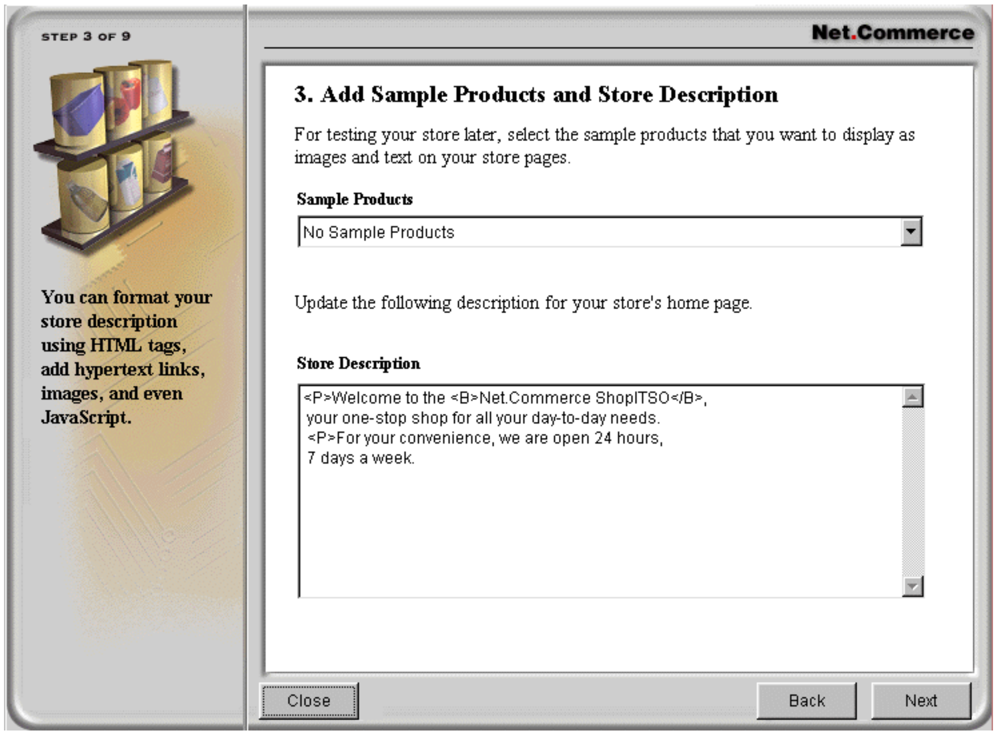
   If you loaded sample data, you can either remove that data using the Product Categories and Product Information forms in Store Manager or you can:

   a. Go to the Store Creator.
   b. Select your store.
   c. Go to step 3.
   d. Change Product Type to "No Product Information".
   e. Go to step 9.
   f. Select **Update**.

   Once the data is customized outside of Store Creator, you should no longer use Store Creator because you may lose changes.

Note

For your first installation in a test environment, it is helpful to select a sample product pallet. Then, you get a shop that has full functionality. You can test your first Net.Data macro changes against this store to see the results. Do not forget to delete these products later from the database.


STEP 3 OF 9

Net.Commerce

3. Add Sample Products and Store Description

For testing your store later, select the sample products that you want to display as images and text on your store pages.

Sample Products

No Sample Products

Update the following description for your store's home page.

Store Description

```
<P>Welcome to the <B>Net.Commerce ShopITSO</B>,
 your one-stop shop for all your day-to-day needs.
 <P>For your convenience, we are open 24 hours,
 7 days a week.
```

You can format your store description using HTML tags, add hypertext links, images, and even JavaScript.

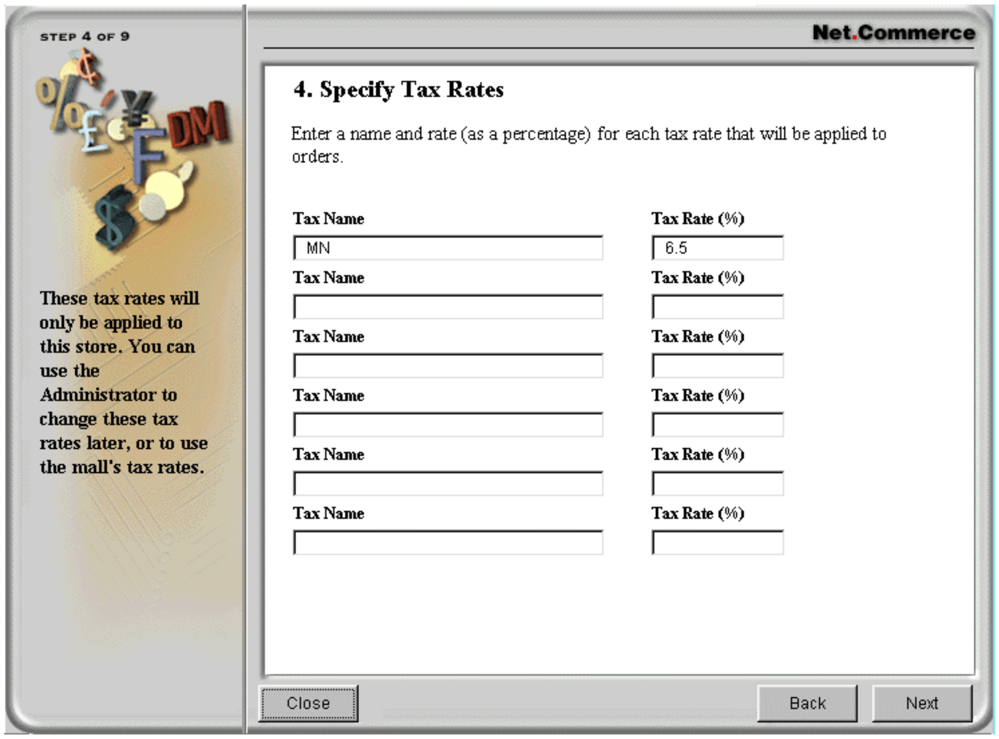
Figure 141. Store Creator — Step 3. Add Sample Products and Store Description

9. Specify tax names and rates, as shown in Figure 142 on page 197, and click the **Next** button.

Note

The tax rates that you enter in this form applies to all items in an order.

If you wish to use flat Tax Rates for all items in an order, enter them in this form. You have the possibility to enter six different tax rates (if you have more than one tax), which are calculated for the whole order. The store tax rates will be stored in the MERCHANTTAX table and will be used instead of the MALL tax rates.


STEP 4 OF 9

These tax rates will only be applied to this store. You can use the Administrator to change these tax rates later, or to use the mall's tax rates.

Net.Commerce

4. Specify Tax Rates

Enter a name and rate (as a percentage) for each tax rate that will be applied to orders.

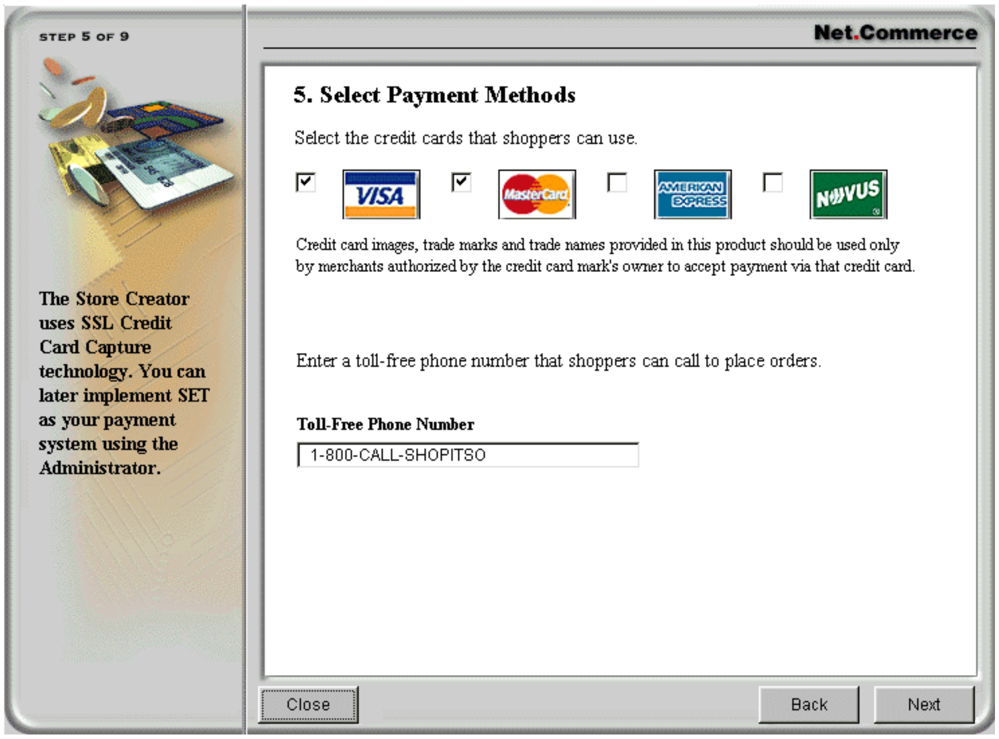
Tax Name	Tax Rate (%)
MN	6.5

Figure 142. Store Creator — Step 4. Specify Tax Rates

10. Select payment methods, as shown in Figure 143 on page 198. Click the **Next** button.

Note

You can select which credit cards will be accepted for basic credit card payments. SET payment will not be configured by Store Creator. If you wish to use it, you have to configure it separately. See 16.1.1, “Installing Payment Server” on page 343, and 16.1.6, “Acquirer Configuration of the Payment Server” on page 353. The selected cards will be listed for the shopper to select during payment.



5. Select Payment Methods

Select the credit cards that shoppers can use.

[✓] VISA [✓] MasterCard [ ] American Express [ ] NOVUS

Credit card images, trade marks and trade names provided in this product should be used only by merchants authorized by the credit card mark's owner to accept payment via that credit card.

Enter a toll-free phone number that shoppers can call to place orders.

Toll-Free Phone Number: 1-800-CALL-SHOPITSO

The Store Creator uses SSL Credit Card Capture technology. You can later implement SET as your payment system using the Administrator.

Figure 143. Store Creator — Step 5. Select Payment Methods

11.Specify shipping providers and charges, as shown in Figure 144 on page 199, and click the **Next** button.

Note

The Shipping Providers (Shipping Service Name) and charges for your store are defined for the entire order. Only flat-rate charges can be defined.

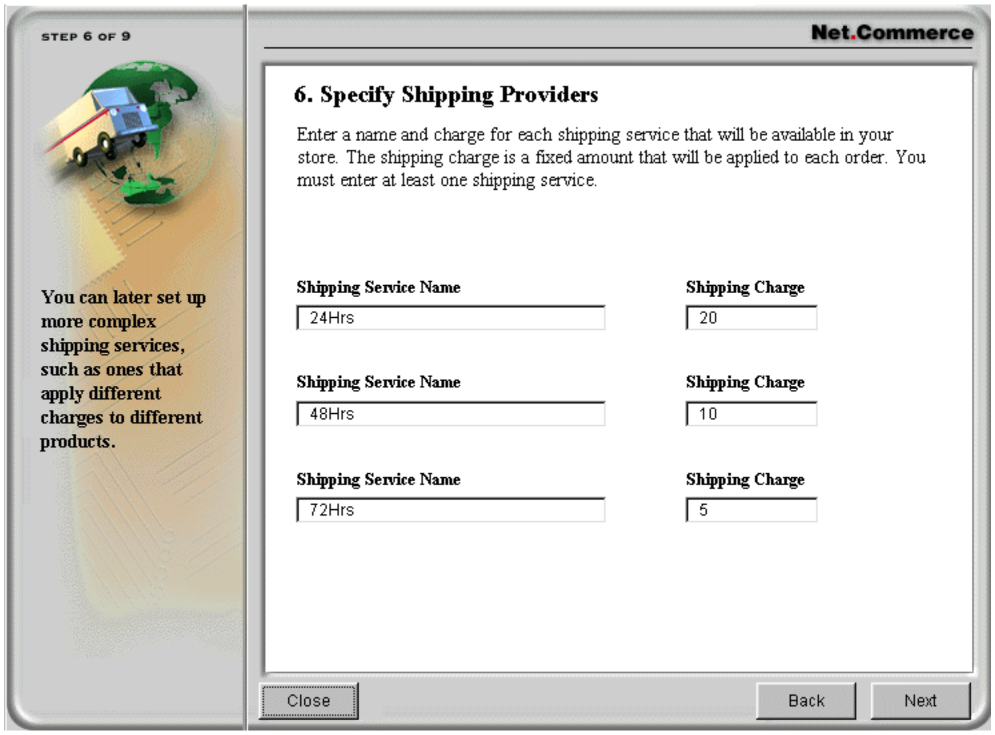
For more complex shipping charge calculations, use the Shipping Services form in the Store Manager function.

Important

To activate the shipping charge calculation for the flat rate changes, you have to assign the value SCSHP to all of your products in your store through the Store Management function. This value has to be chosen in the Product Shipping Code selection list of the Product/Item Information form, that you get when you choose the Product Information button in the left site of the Store Manager window. See Figure 153 on page 208.

To avoid this manual entry for all of your products, you can use an SQL update query to set the value in the field PRPSNBR of the PROCUCT table. Be careful. The value is than *not* SCSHP as mentioned before. It must be the value of the merchant reference number that will be created, when you are finished with the creation of the store through the Store Creator. You can find this value in the define variable MerchantRefNum in the include file for this store. The include file is in the directory `/QIBM/UserDataNetCommerce/instance/instance_name/macro/Shop_name/`. Also the merchant reference number is stored in the merchant table MERFNBR field, where you can find it.

It is also possible to set the value for the PRPSNBR field in the product table through the mass import function, after the store is created. Use the PRSCODE parameter with the #PRODUCT command and assign the merchant reference number.



6. Specify Shipping Providers

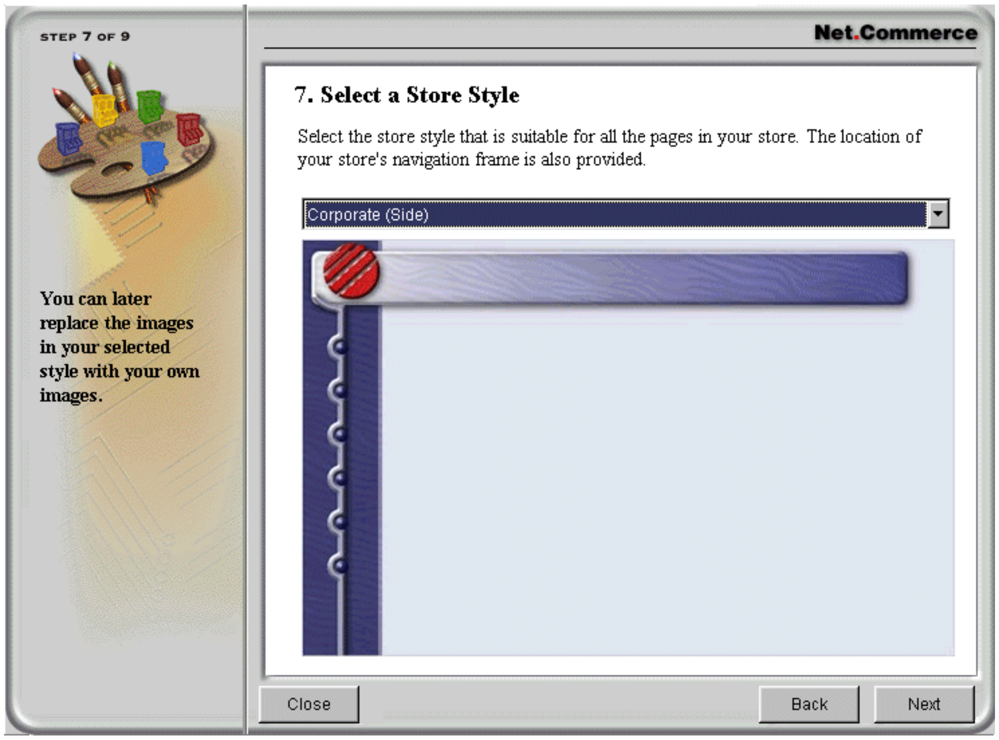
Enter a name and charge for each shipping service that will be available in your store. The shipping charge is a fixed amount that will be applied to each order. You must enter at least one shipping service.

You can later set up more complex shipping services, such as ones that apply different charges to different products.

Shipping Service Name	Shipping Charge
24Hrs	20
48Hrs	10
72Hrs	5

Figure 144. Store Creator — Step 6. Specify Shipping Providers

12.Select **Corporate** for the store style, as shown in Figure 145 on page 200, and click the **Next** button.


STEP 7 OF 9

Net.Commerce

7. Select a Store Style

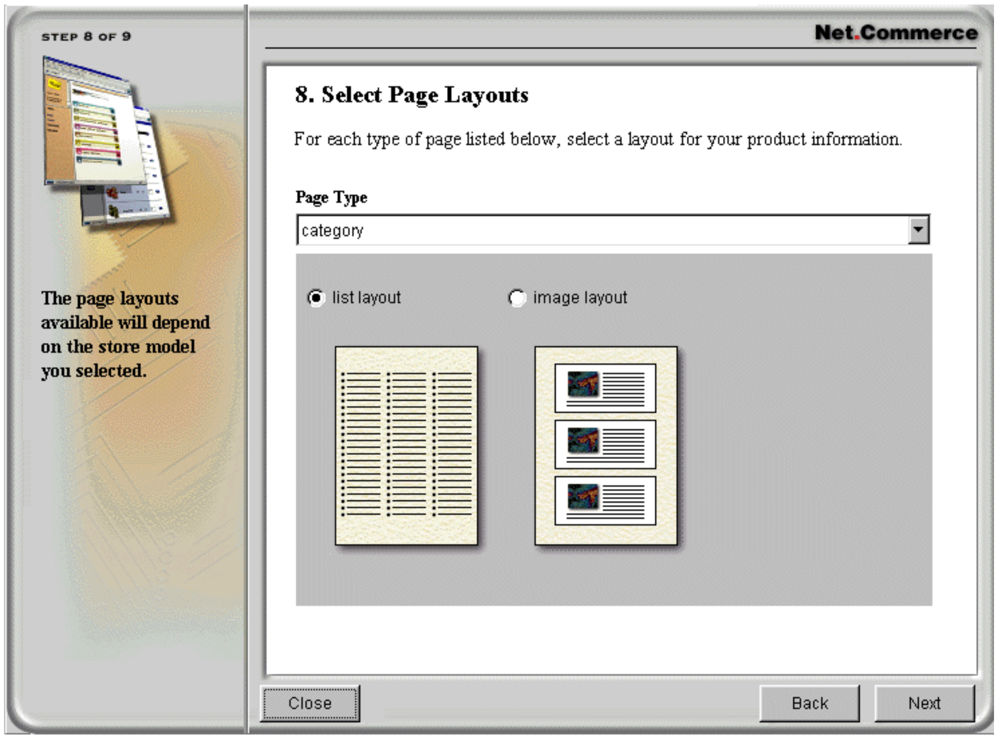
Select the store style that is suitable for all the pages in your store. The location of your store's navigation frame is also provided.

Corporate (Side)

You can later replace the images in your selected style with your own images.

Figure 145. Store Creator — Step 7. Select a Store Style

13.Select **list layout** for the category pages, as shown in Figure 146. Click the **Next** button.


STEP 8 OF 9

Net.Commerce

8. Select Page Layouts

For each type of page listed below, select a layout for your product information.

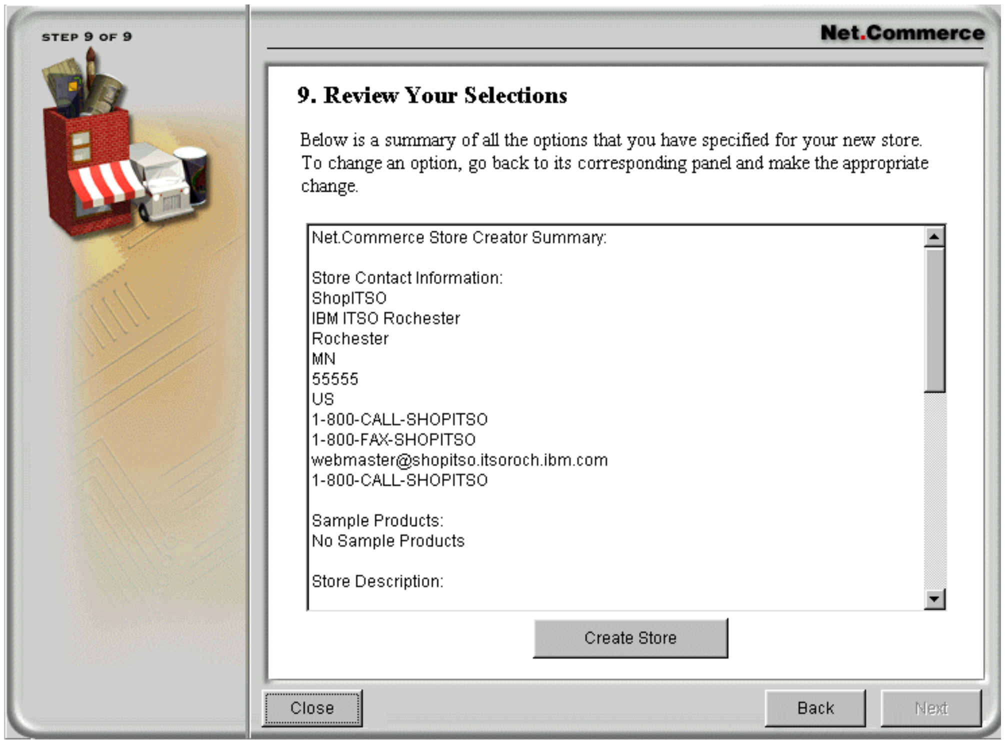
Page Type: category

[✓ list layout] image layout

The page layouts available will depend on the store model you selected.

Figure 146. Store Creator — Step 8. Select Page Layouts

14.Review your selections, as shown in Figure 147, and click **Create Store** to create the store.



9. Review Your Selections

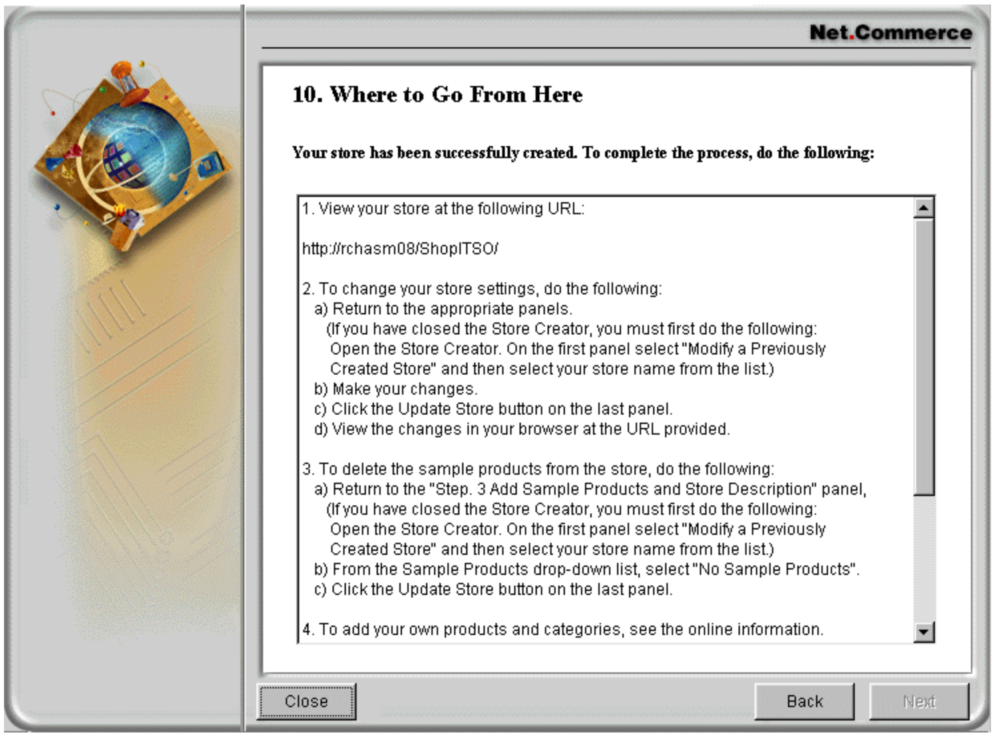
Below is a summary of all the options that you have specified for your new store. To change an option, go back to its corresponding panel and make the appropriate change.

```
Net.Commerce Store Creator Summary:

Store Contact Information:
ShopITSO
IBM ITSO Rochester
Rochester
MN
55555
US
1-800-CALL-SHOPITSO
1-800-FAX-SHOPITSO
webmaster@shopitso.itsoroch.ibm.com
1-800-CALL-SHOPITSO

Sample Products:
No Sample Products

Store Description:
```

Figure 147. Store Creator — Step 9. Review Your Selections

15.After the creation process is finished, the results window, shown in Figure 148 on page 202, describes how to access the new store.

Click **Close** to close the Store Creator window. The Store Creator Status window closes automatically.



10. Where to Go From Here

Your store has been successfully created. To complete the process, do the following:

```
1. View your store at the following URL:

http://rchasm08/ShopITSO/

2. To change your store settings, do the following:
  a) Return to the appropriate panels.
    (If you have closed the Store Creator, you must first do the following:
     Open the Store Creator. On the first panel select "Modify a Previously
     Created Store" and then select your store name from the list.)
  b) Make your changes.
  c) Click the Update Store button on the last panel.
  d) View the changes in your browser at the URL provided.

3. To delete the sample products from the store, do the following:
  a) Return to the "Step. 3 Add Sample Products and Store Description" panel,
    (If you have closed the Store Creator, you must first do the following:
     Open the Store Creator. On the first panel select "Modify a Previously
     Created Store" and then select your store name from the list.)
  b) From the Sample Products drop-down list, select "No Sample Products".
  c) Click the Update Store button on the last panel.

4. To add your own products and categories, see the online information.
```

Figure 148. Store Creator — Results Display

16. Add the Pass directives for the new store in the Web Server Configuration File of the HTTP instance.

```
Pass /QIBM/UserData/NetCommerce/instance/instance_name/html/shop_name
```

The pass directives for our sample ShopITSO are:

```
Pass /shopitso/* /QIBM/UserData/NetCommerce/instance/work/html/ShopITSO/*
Pass /shopITSO/* /QIBM/UserData/NetCommerce/instance/work/html/ShopITSO/*
Pass /SHOPITSO/* /QIBM/UserData/NetCommerce/instance/work/html/ShopITSO/*
```

With these three pass directives, the customer can work with all three variations of the URL: `http://myServer/shopitso`, `http://myServer/shopITSO`, and `http://myServer/SHOPITSO`.

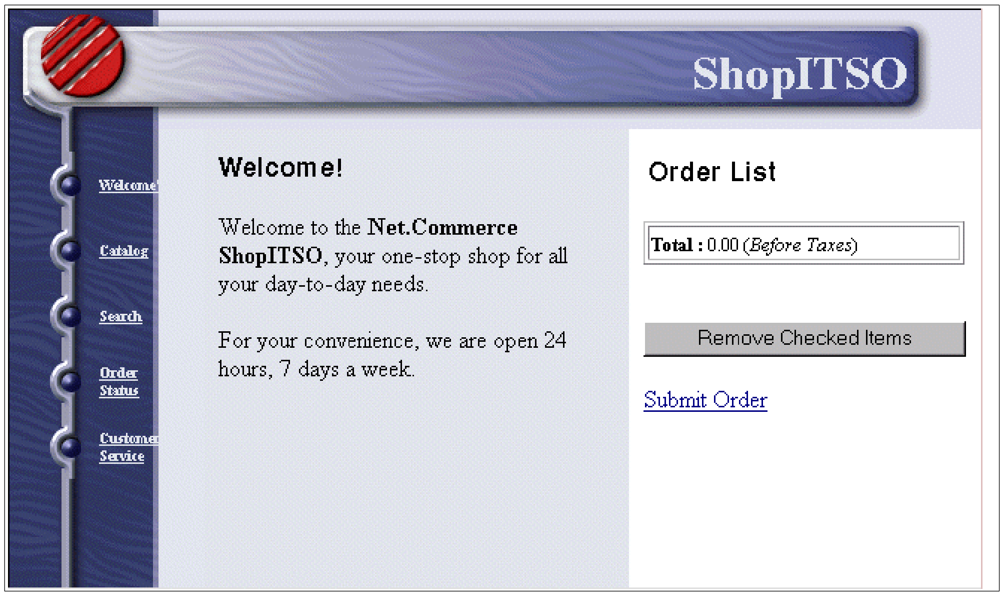
Section A.11, “AS/400 Web Server Configuration File” on page 519, shows our configuration file with which we worked.

17. Stop and restart the HTTP server through the HTTP server configuration function in the AS/400 task or use following AS/400 commands from a 5250 session:

```
ENDTCPSVR SERVER(*HTTP) HTTPSVR(instance_name)
STRTCPSVR SERVER(*HTTP) HTTPSVR(instance_name)
```

18. Finally, to access the new store, open a browser and follow the URL link indicated in Figure 148.

The resulting store, previous to our customizing, is shown in Figure 149 on page 203.



Welcome!

Welcome to the **Net.Commerce ShopITSO**, your one-stop shop for all your day-to-day needs.

For your convenience, we are open 24 hours, 7 days a week.

Total : 0.00 (*Before Taxes*)

Figure 149. Shop ITSO Just after Creation by Store Creator

After creating the store with the Store Creator, you need to create the product catalog, see 13.7, “Creating the Product Catalog” on page 203.

A root category named Top Category has been created for you. You need to create additional categories and the products under this root.

13.7 Creating the Product Catalog

Figure 150 summarizes the steps for creating the Product Catalog in the Net.Commerce e-business application. This section describes, in detail, how we implemented the Product Catalog for our sample store ShopITSO.

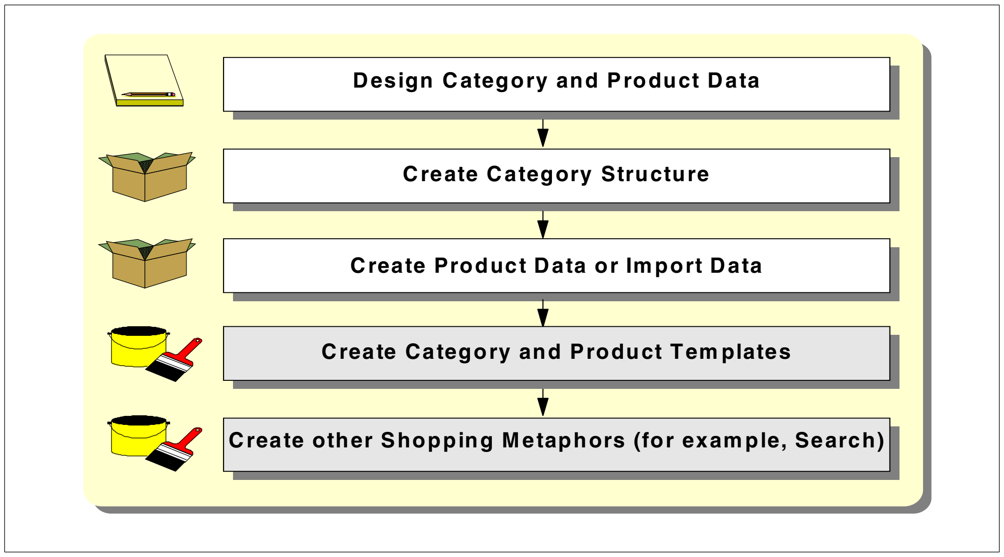

Design Category and Product Data
Create Category Structure
Create Product Data or Import Data
Create Category and Product Templates
Create other Shopping Metaphors (for example, Search)

Figure 150. Creating the Product Catalog

1. Design Category and Product Data.

   To learn how the category and product data in Net.Commerce are defined and work, see 3.2, “Planning the Product Catalog” on page 31.

2. Create Category Structure.

   First, we imported our products with the mass import function (see 15.3.2, “The LOADPRD Utility — Description” on page 330) above the root category named Top Category, which was build by the Store Creator.

   Our mass import function from Net.Commerce imported only one category with the name Top Category. All products belong to this category.

   After the mass import, we used the Store Manager to build the hierarchy and add new categories. See 13.12, “Customizing the Category Tree” on page 222.

3. Create Product Data or Import Data.

   We used the mass import function to import the products from the back-end system to the Net.Commerce database together with the category. See point 2 above.

   Before the mass import step, we proved our production databases to find out which characteristics in our production tables can be used to build the mass import file for the mass import. We found out that our tables already have some characteristics to build the content for the Net.Commerce database tables. For more information about mass import, see 15.3, “Importing Data by Example” on page 330.

   Table 12 on page 205 shows an extract from the matching table that we built. It points out which tables and fields are used from our production system to fill the mass import file.

   In our sample store, we have only small text for the two long description fields in the product table. We imported this text from our back-end system table BELDSC (see Table 12).

   In a real store, you have to import the values for these two fields from an HTML source file. The reason is that the contents of these fields are used to show them later in the browser and therefore, should be in the HTML format (see also in 3.2.3, “Planning Product Descriptions” on page 35). This can also be done in a second mass import step.

   A second approach to have the long description available for a product is to use the PRURL field in the product table. In this field, you enter a URL path. In your product template, you can use this URL to make a link to an HTML file that has the description for the product. For a sample of how this works, see 13.11, “Using the Product PRURL Field” on page 218.

Table 12. Fields Used from the Back-End System Table for Product Catalog

Prod.Table Name	Field in Prod. Table or Constant Value	Import File Command and Used Field Name	Fills Net.Commerce Table	Net.Commerce Description
BECATEG	BETEXT	#CATEGORY CGNAME	CATEGORY	Category name
	BETEXT	#CATEGORY CGSDESC	CATEGORY	Short Description
	BETEXT	#CATEGORY CGLDESC	CATEGORY	Long Description
	BETEXT	#CATEGORY Parent_cat_na me	CGPRREL	used to build the reference number for parent category
	Constant: 1	#CATEGORY CRSEQNBR	CGPRREL	CRSEQNBR
	BETEXT	#CATESGP cat-name	CATESGP	used to build the reference number for category
	Constant /ShopITSO/c at1.d2w	#CATESGP CSDISPLAY	CATESGP	CSDISPLAY
BEPROD	BEPNBR	#PRODUCT PRNBR	PRODUCT	PRNBR
	BESDSC	#PRODUCT PRSDESC	PRODUCT	PRSDESC
	BELDSC	#PRODUCT PRLDESC1	PRODUCT	PRLDESC1

4. Create Category and Product Templates.

   In ShopITSO, we used the Net.Data macros cat1.d2w for all of our categories and prod1.d2w for all our products, which were created by the Store Creator. Also, our mass import file used these names to fill the fields in the above mentioned template tables.

   Perhaps you do not have such a simple solution because you work with multiple templates and with multiple shopping groups. These particular cases require more effort to complete the assignment and to create more than one template for a product or category. See 3.2.4, “Planning Category and Product Templates” on page 36, for more information. To learn how you can assign a Net.Data macro template to a category or product without the mass import, see 13.8, “Assigning Templates” on page 206.

5. Create other Shopping Metaphors.

   We use the Product Advisor functionality to provide product comparison and product exploration pages. See Chapter 14, “Enhancing the Store Using Product Advisor” on page 275.

13.8 Assigning Templates

As mentioned in 3.2.4, “Planning Category and Product Templates” on page 36, fill the assigned template path and name through the mass import function. In this section, we show you how to update this information for a single product or category. Both the assignment for categories and product templates are possible through the Store Manager function.

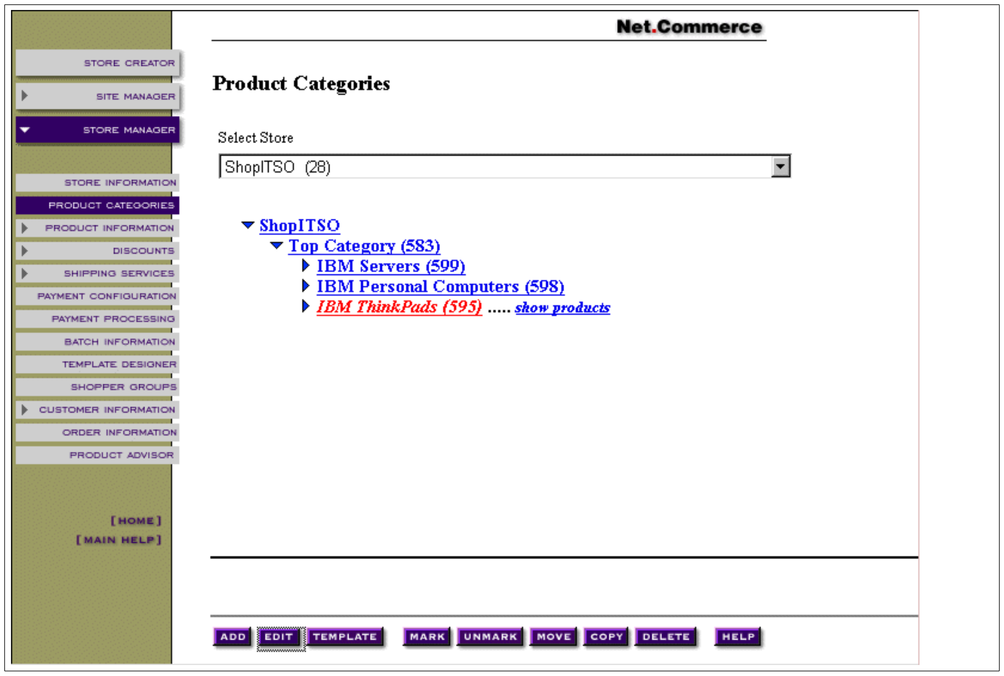
13.8.1 Assigning Templates to Categories

To assign templates to *categories,* use the Store Manager and follow these steps:

1. Login as a Net.Commerce administrator with access to the store whose information you are changing in the Net.Commerce Administration Web page at the following URL:

   http://*fullyqualifiedhost_name*/ncadmin/

2. Select **Product Categories** in the left frame of the window.
3. Select **ShopITSO** (your store name) in the Store selection list.
4. Click on the category you want to assign or prove. See Figure 151.



Product Categories

Select Store

ShopITSO (28)

ShopITSO
Top Category (583)
IBM Servers (599)
IBM Personal Computers (598)
IBM ThinkPads (595) ..... show products

Figure 151. Product Categories

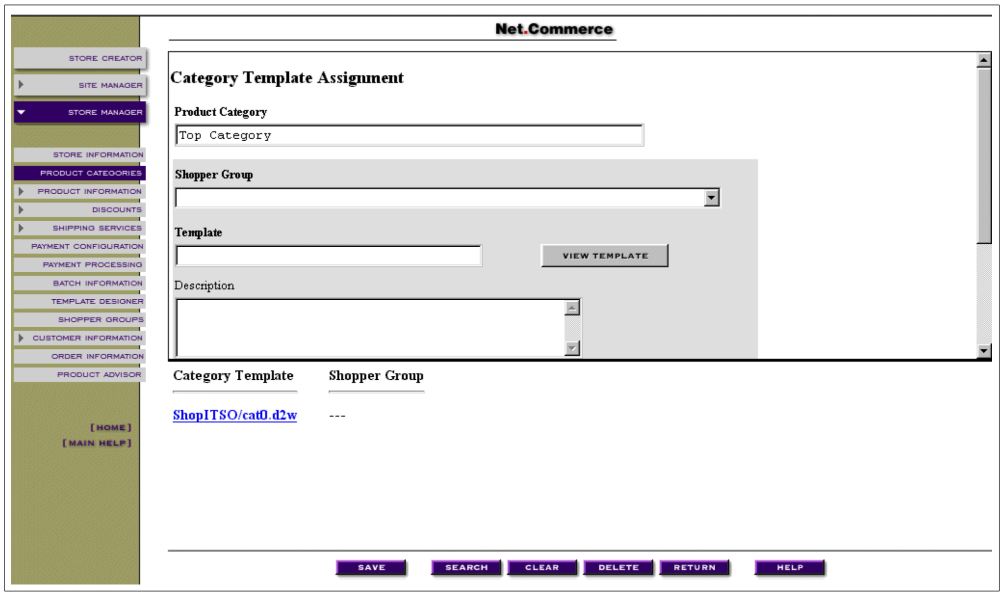
5. Click the **TEMPLATE** button.
6. Click the **SEARCH** button.

Tip

To obtain a larger second half of the screen, set the cursor of the black line and move this line above.

Now you can see the assigned template path ShopITSO name cat0.d2w for the chosen category, as shown in Figure 152.

7. In this window, make your changes as required. To learn how to add a template for a new category, see 13.12, "Customizing the Category Tree" on page 222. Begin with Figure 178 on page 224.



Category Template Assignment

Product Category
Top Category

Shopper Group

Template

Description

Category Template	Shopper Group
ShopITSO/cat0.d2w	---

Figure 152. Assigned Category Template

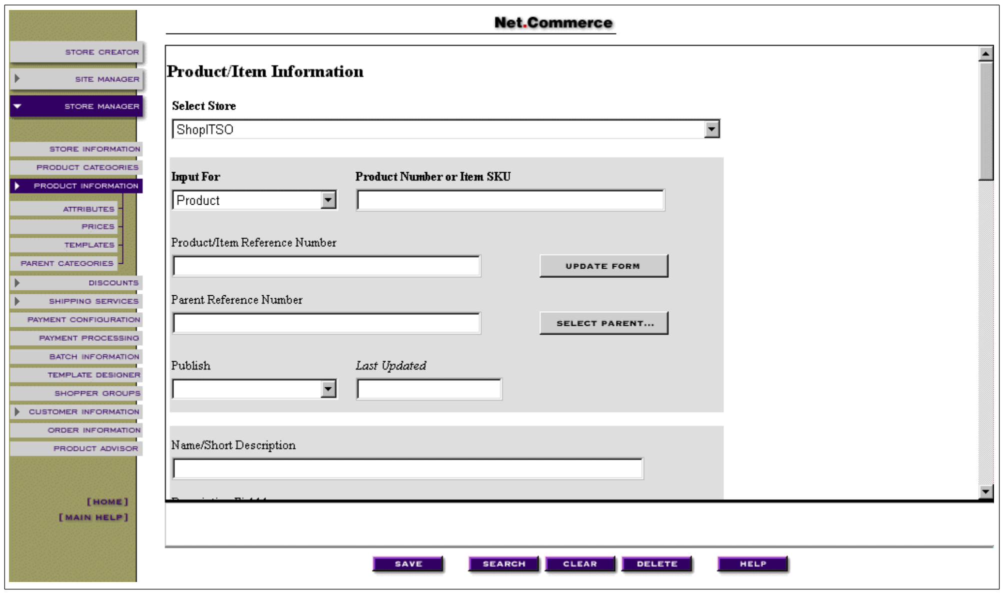
13.8.2 Assigning Templates to Products

To assign templates to *products*, use the Store Manager and follow these steps:

1. Login as `ncadmin` in the Net.Commerce Administration Web page at the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

2. Select **Product Information** in the left frame of the window.
3. Select **Store ShopITSO** (your store name) in the Store selection list.
4. Choose **Product** or **Item** in the Input For selection list. See Figure 153 on page 208.



Product/Item Information

Select Store: ShopITSO

Input For: Product

Product Number or Item SKU:

Product/Item Reference Number:

Parent Reference Number:

Publish:

Last Updated:

Name/Short Description:

Figure 153. Product/Item Information

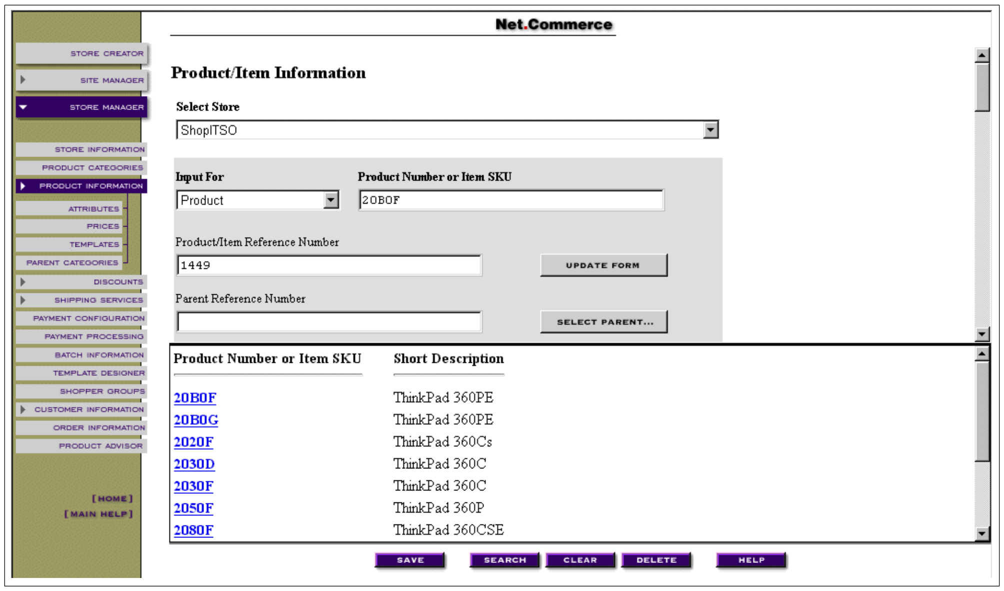
5. Click the **SEARCH** button.

Tip

To obtain a larger second half of the screen, set the cursor of the black line and move this line above.

6. Click on the product or item you want to change.

   The chosen product or item is shown in the Product Number or Item SKU field. See Figure 154.



Product/Item Information

Select Store: ShopITSO

Input For: Product

Product Number or Item SKU: 20B0F

Product/Item Reference Number: 1449

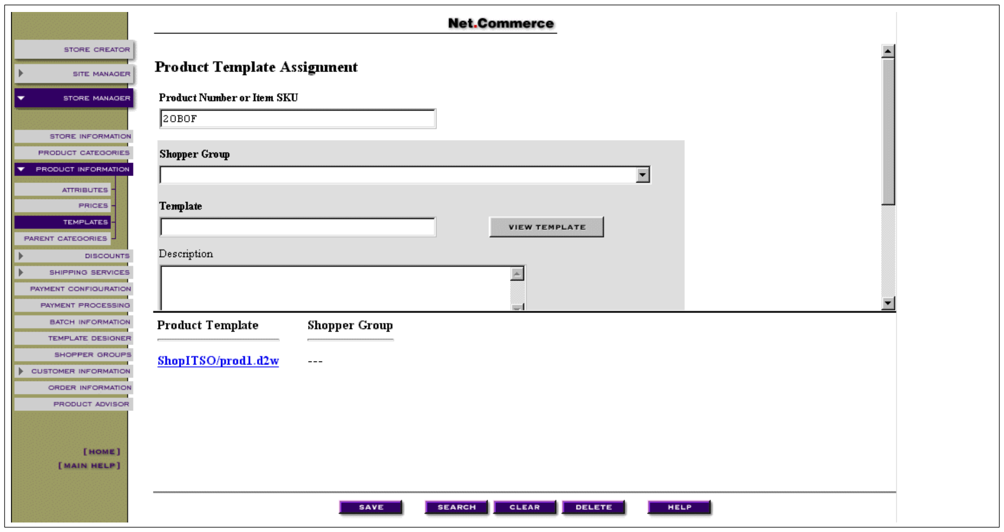
Parent Reference Number:

Product Number or Item SKU	Short Description
20B0F	ThinkPad 360PE
20B0G	ThinkPad 360PE
2020F	ThinkPad 360Cs
2030D	ThinkPad 360C
2030F	ThinkPad 360C
2050F	ThinkPad 360P
2080F	ThinkPad 360CSE

Figure 154. Product/Item Information — Product List

7. Select **TEMPLATES** in the left frame of the window.
8. Select **None (default template)** in the Shopper Group selection list.
9. Push the **SEARCH** button.

   Now you can see the assigned template path ShopITSO name prod1.d2w for the chosen product like in Figure 155. The Shopper Group field is blank. This means this template is not assigned to a shopper group.



Product Template Assignment

Product Number or Item SKU: 20BOF

Shopper Group:

Template:

Description:

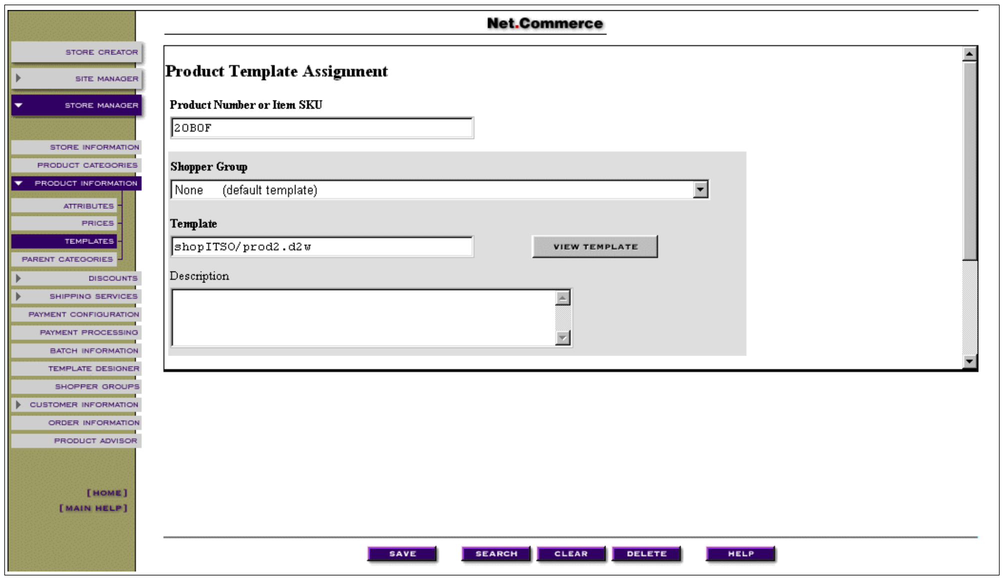
Product Template	Shopper Group
ShopITSO/prod1.d2w	---

Figure 155. Product Template Assignment

10. Make your changes as required here:
    a. Select Shopper Group **None (default template)** in the selection list.
    b. Type in the path and name of the template as:

    ```
    ShopITSO/prod1.d2w
    ```

    c. Push the **SAVE** button. Figure 156 shows these changes.



Product Template Assignment

Product Number or Item SKU: 20BOF

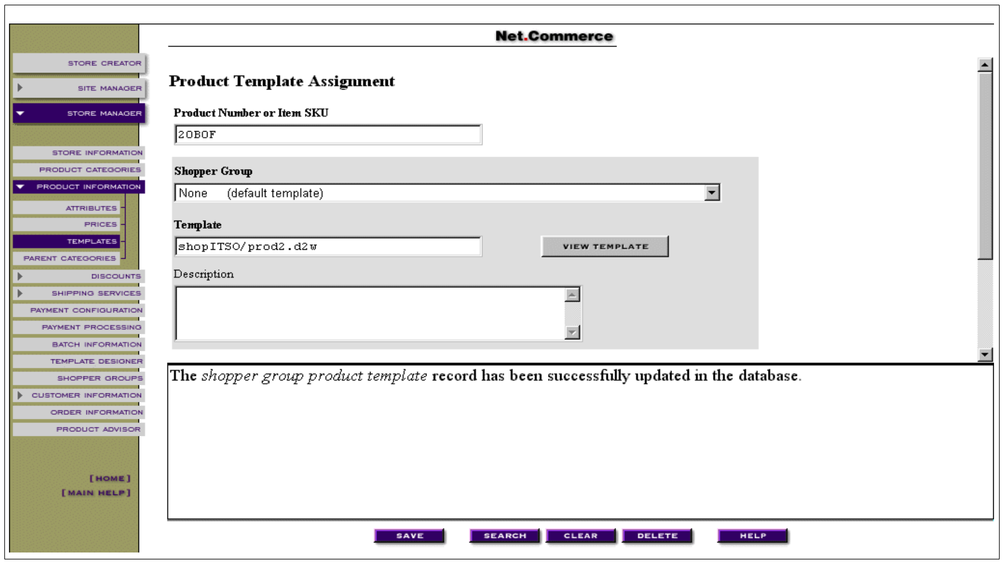
Shopper Group: None (default template)

Template: shopITSO/prod2.d2w

Description:

Figure 156. Update Template Assignment

If the update completed successfully, a message appears as shown in Figure 157.



Product Template Assignment

Product Number or Item SKU: 20BOF

Shopper Group: None (default template)

Template: shopITSO/prod2.d2w

Description:

The *shopper group product template* record has been successfully updated in the database.

Figure 157. Product Template Updated Successfully

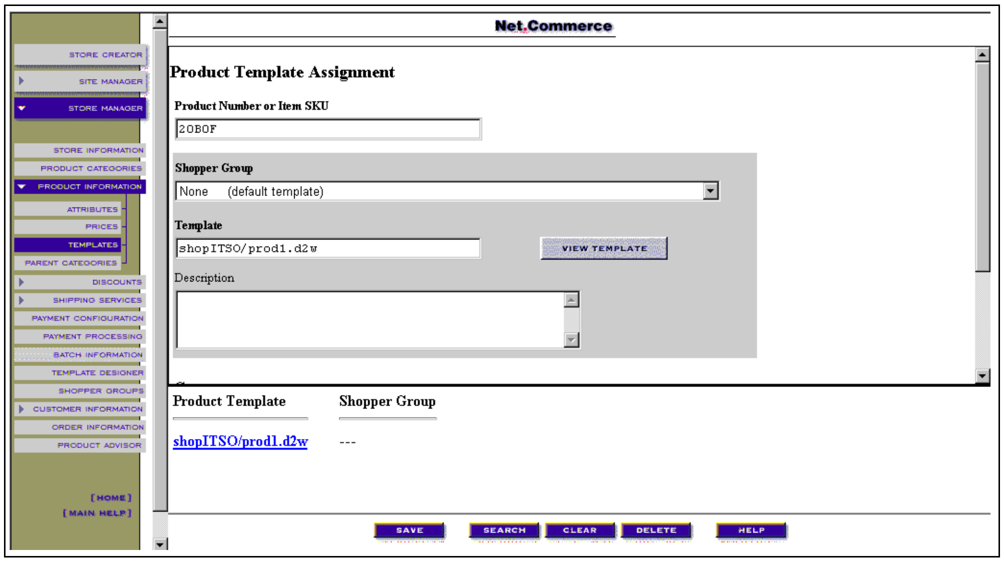
13.8.3 Quick Test for Template Views

For a quick look at how your product or category page will appear, you can use the following approach for a product template. To look for a category template page, the steps are similar, with the exception that you start with the Product Categories form from the Store manager. See 13.8.1, “Assigning Templates to Categories” on page 206.

To view a window where your can see how your page will appear, complete the following steps:

1. Follow steps 1 through 11, which are described in 13.8.2, “Assigning Templates to Products” on page 207.
2. Click on the template you want to see.

   A window appears similar the one shown in Figure 158 on page 211.



Product Template Assignment

Product Number or Item SKU: 20B0F

Shopper Group: None (default template)

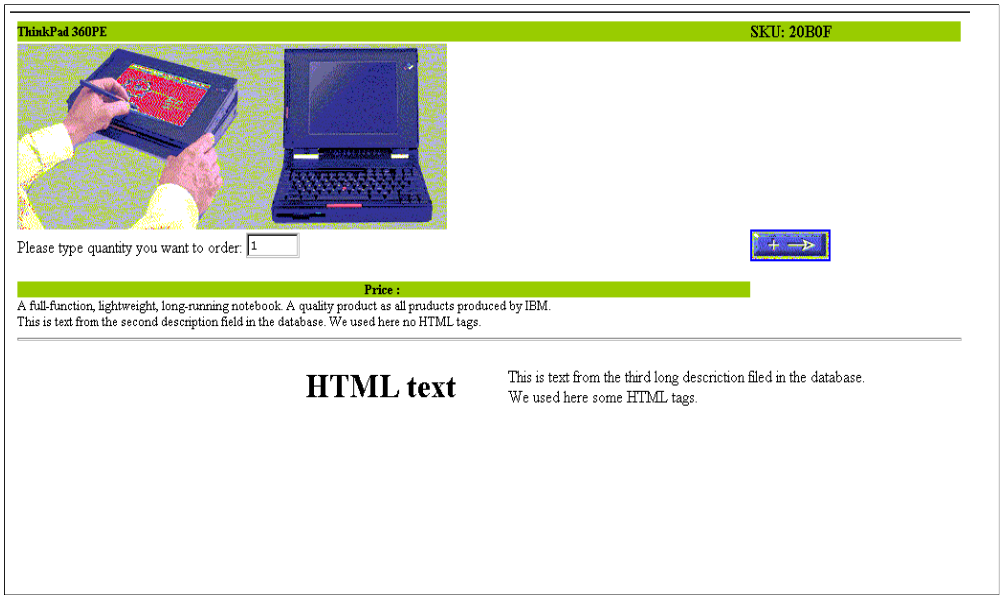
Template: shopITSO/prod1.d2w

Description

Product Template	Shopper Group
shopITSO/prod1.d2w	---

Figure 158. Product Template Assignment

3. Click the **VIEW TEMPLATE** button. The test template for the chosen product appears (Figure 159).



ThinkPad 360PE SKU: 20B0F

Please type quantity you want to order: 1

Price :

A full-function, lightweight, long-running notebook. A quality product as all pruducts produced by IBM.
This is text from the second description field in the database. We used here no HTML tags.

HTML text

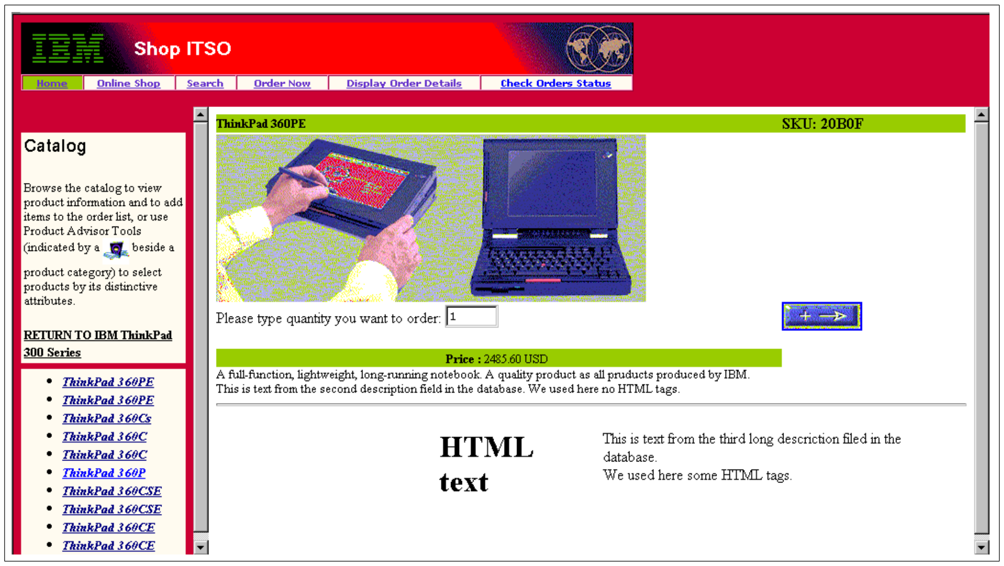
This is text from the third long descriction filed in the database.
We used here some HTML tags.

Figure 159. Test View from the Product Template

Note

Because this view is only built with the Net.Date macro information, you cannot see the frames around the page as in your real e-business application.

Figure 160 on page 212 shows our product page with the same product as shown on Figure 159 in the real ShopITSO application. Notice that now the height of the product page in the main frame does not fit the browser window size, and a scroll bar appears on the right side of the main frame.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

ThinkPad 360PE SKU: 20B0F

Please type quantity you want to order: 1

Price : 2485.60 USD

A full-function, lightweight, long-running notebook. A quality product as all pruducts produced by IBM.
This is text from the second description field in the database. We used here no HTML tags.

HTML text

This is text from the third long descriction filed in the database.
We used here some HTML tags.

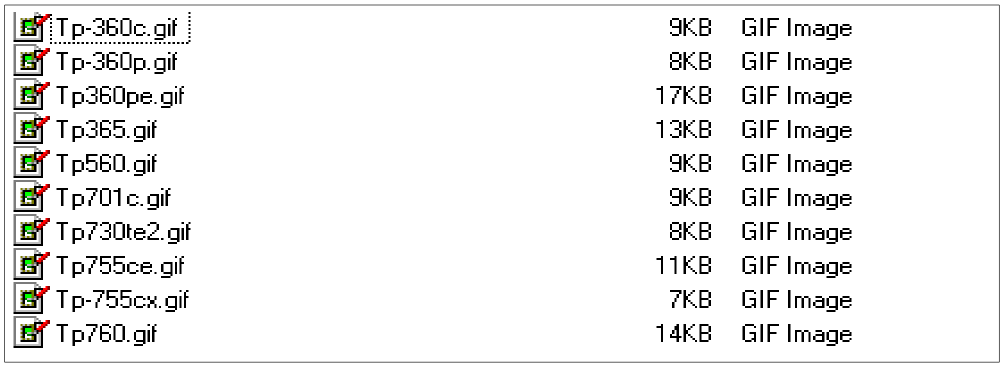
Figure 160. Product Page in ShopITSO

13.9 Assigning Product Images to Products

The best way to store the image path and name in the Net.Commerce PRODUCTS database table is to use the mass import function. See 3.3, “Images and Multimedia Files” on page 37, and 15.2.2, “Mass Import” on page 321. You can also use the Store Manager function to update both the PRTHMB and PRFULL image fields for a single product.

In our case, our mass import file did not fill this field for the PRODUCT table. We did this intentionally to show you another way of how you can solve this problem. After we received the product data from the mass import, we used interactive SQL statements to make our changes. Because we are using the same image file for many similar products, the SQL technique allows us to update many records with a single SQL command. If you do not have Interactive SQL installed on your system, you should generate mass import records and use the mass import utility to update your product records.

In our ShopITSO, we have the following images for products available. See Figure 161 on page 213. The image names have a short form of our product names field PRSDESC in the Net.Commerce table product.



```
Tp-360c.gif      9KB   GIF Image
Tp-360p.gif      8KB   GIF Image
Tp360pe.gif     17KB   GIF Image
Tp365.gif       13KB   GIF Image
Tp560.gif        9KB   GIF Image
Tp701c.gif       9KB   GIF Image
Tp730te2.gif     8KB   GIF Image
Tp755ce.gif     11KB   GIF Image
Tp-755cx.gif     7KB   GIF Image
Tp760.gif       14KB   GIF Image
```

Figure 161. Images Available for Products

We used the following SQL statement to see our product names in the products table for our shop (merchant number is 28):

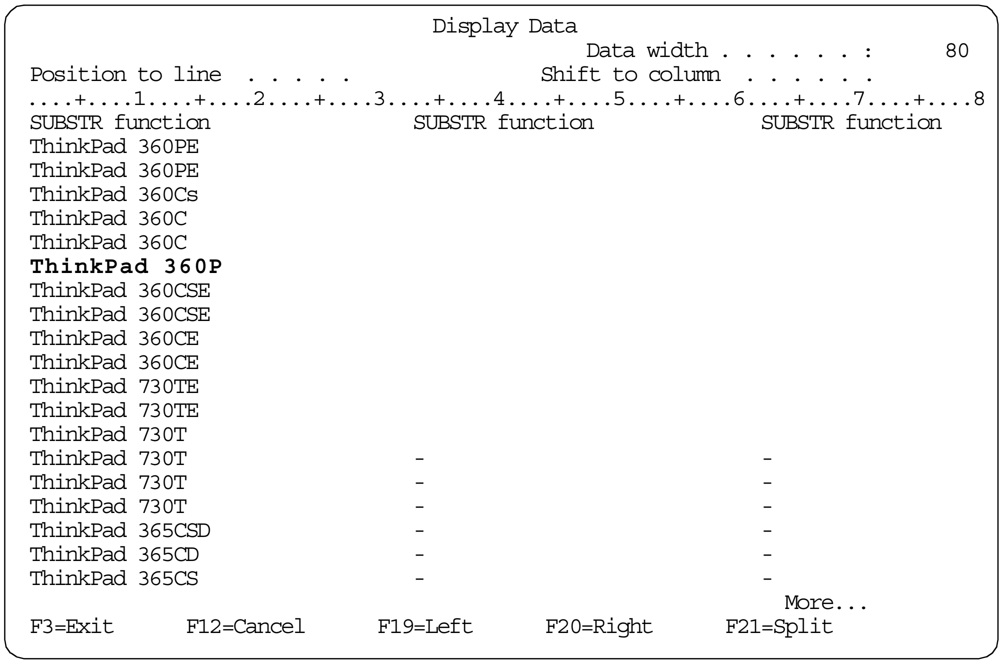
```
SELECT SUBSTR(PRSDESC, 1,30), SUBSTR(PRTHMB, 1,27), SUBSTR(PTFULL, 1,27)
FROM WORK/PRODUCT
WHERE PRMENBR=28
```

Figure 162 shows the results of this SQL query.

Note

Use the AS/400 command `ADDLIBLE QNETCOMM` to add the Net.Commerce program library to your library list before you use interactive SQL. This library contains the trigger programs that take care of the integrity of your database.

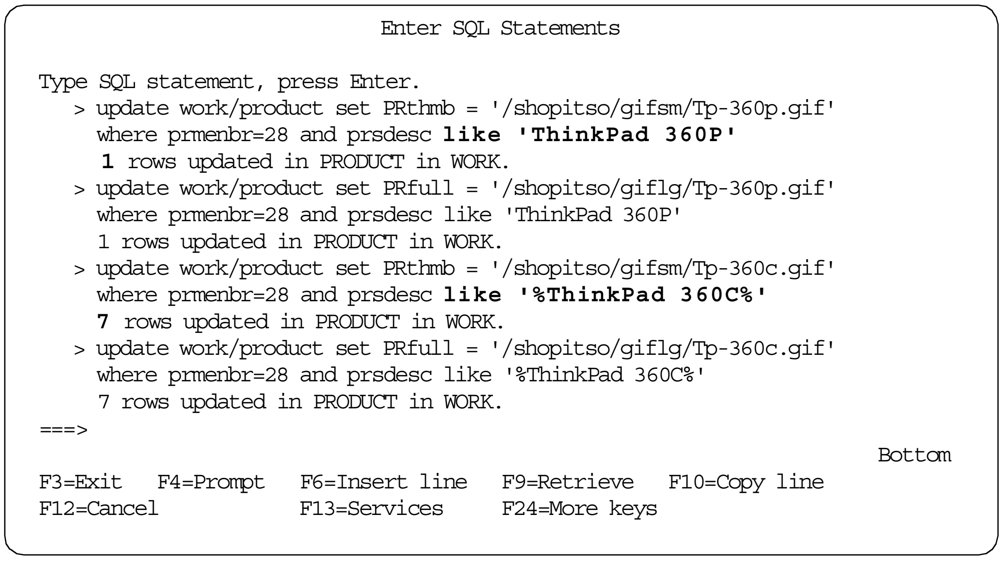
If you do not add this library, you receive the error message: `Trigger program or external procedure detected an error.`



```
                                  Display Data
                                               Data width . . . . . . :      80
Position to line  . . . . .                Shift to column  . . . . . .
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
SUBSTR function                 SUBSTR function             SUBSTR function
ThinkPad 360PE
ThinkPad 360PE
ThinkPad 360Cs
ThinkPad 360C
ThinkPad 360C
ThinkPad 360P
ThinkPad 360CSE
ThinkPad 360CSE
ThinkPad 360CE
ThinkPad 360CE
ThinkPad 730TE
ThinkPad 730TE
ThinkPad 730T
ThinkPad 730T                   -                           -
ThinkPad 730T                   -                           -
ThinkPad 730T                   -                           -
ThinkPad 365CSD                 -                           -
ThinkPad 365CD                  -                           -
ThinkPad 365CS                  -                           -
                                                              More...
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split
```

Figure 162. Our Product Names

The next step was to assign the image names to the product. Because of our name rule, this was not a big effort. Therefore, it was easy to build SQL statements for updating the product table.



```
                              Enter SQL Statements

Type SQL statement, press Enter.
     > update work/product set PRthmb = '/shopitso/gifsm/Tp-360p.gif'
       where prmenbr=28 and prsdesc like 'ThinkPad 360P'
        1 rows updated in PRODUCT in WORK.
     > update work/product set PRfull = '/shopitso/giflg/Tp-360p.gif'
       where prmenbr=28 and prsdesc like 'ThinkPad 360P'
       1 rows updated in PRODUCT in WORK.
     > update work/product set PRthmb = '/shopitso/gifsm/Tp-360c.gif'
       where prmenbr=28 and prsdesc like '%ThinkPad 360C%'
       7 rows updated in PRODUCT in WORK.
     > update work/product set PRfull = '/shopitso/giflg/Tp-360c.gif'
       where prmenbr=28 and prsdesc like '%ThinkPad 360C%'
       7 rows updated in PRODUCT in WORK.
===>
                                                                    Bottom
F3=Exit   F4=Prompt   F6=Insert line   F9=Retrieve   F10=Copy line
F12=Cancel            F13=Services     F24=More keys
```

Figure 163. SQL Statements for Updating the Image Path and Names

The SQL statement `like 'ThinkPad 360P'` only updated one row. The product with description ThinkPad 360 PE was not updated (this product should have the image Tp360Pe assigned). The SQL statement `like '%ThinkPad 360C%'` made the update for all products with the name ThinkPad 360C, ThinkPad 360CS, and ThinkPad 360CSE. We repeated this last step for all our products.

13.10 Using Product Long Description Fields

As mentioned in 3.2.3, “Planning Product Descriptions” on page 35, you should complete the necessary product descriptions throughout the mass import function. In this section, we show you how you can update this information for a single product.

To make any changes that are related to the several product tables in Net.Commerce, you can use the Store Manager product forms. This section describes how you can change the information for the product long description fields. To change the description for *products*, use the Store Manager and follow these steps:

1. Login as `ncadmin` in the Net.Commerce Administration Web page at the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

2. Select **Product Information** in the left frame of the window.
3. Select **Store ShopITSO** (your store name) in the Store selection list.
4. Choose **Product** or **Item** in the Input For selection list. See Figure 153 on page 208.
5. Click the **SEARCH** button.

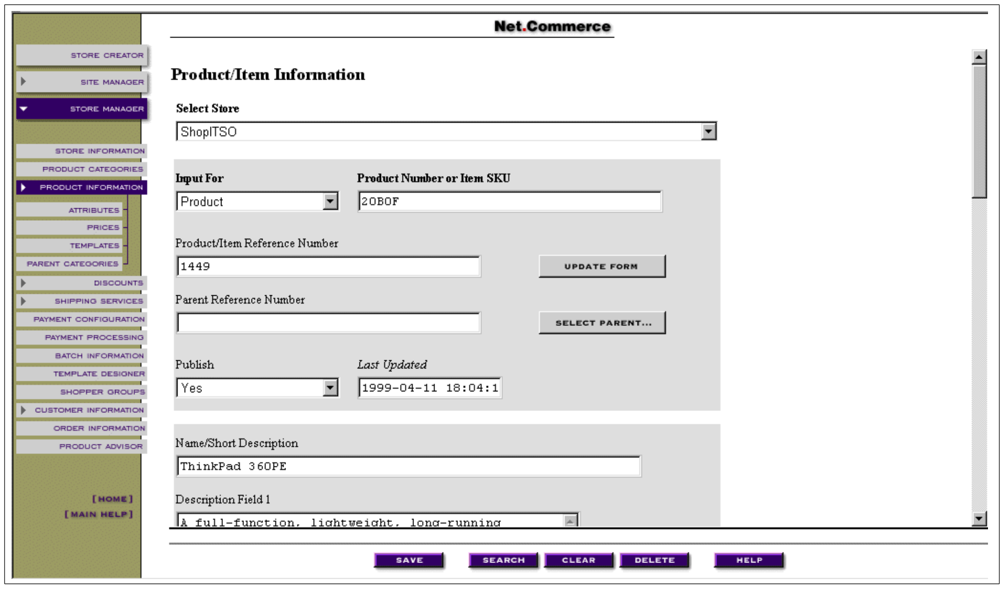
Tip

To get a larger second half of the screen, set the cursor of the black line and move this line above.

6. Click on the product or item you want to change.
7. The chosen product or item is shown in the Product Number or Item SKU field. You can see also all fields that you can update. See Figure 164.

Tip

Make the second part of the window smaller to see the fields you can change.



Product/Item Information

Select Store: ShopITSO

Input For: Product

Product Number or Item SKU: 20B0F

Product/Item Reference Number: 1449

Parent Reference Number:

Publish: Yes

Last Updated: 1999-04-11 18:04:1

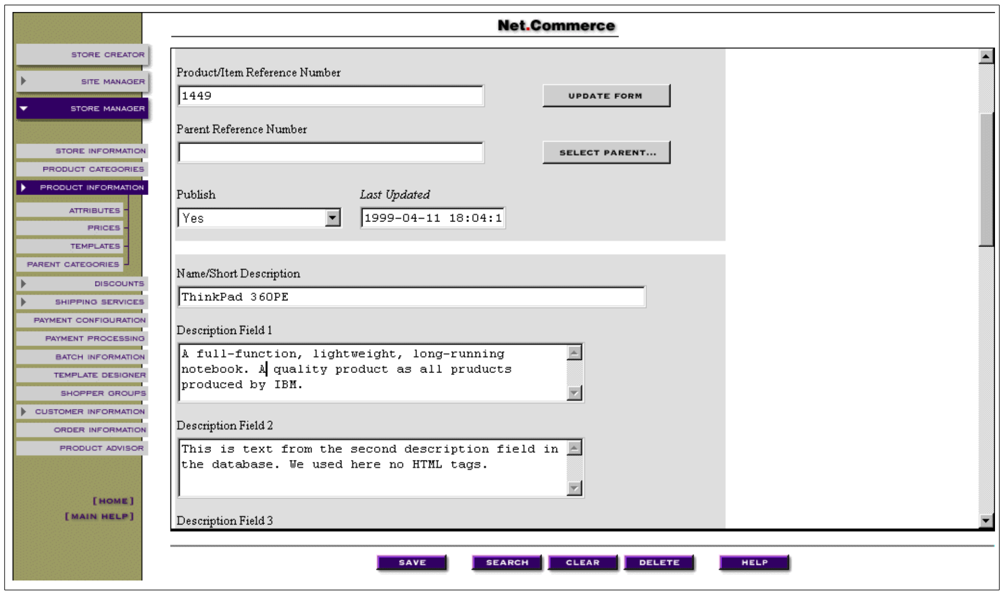
Name/Short Description: ThinkPad 360PE

Description Field 1: A full-function, lightweight, long-running

Figure 164. Product/Item Information (Part 1 of 6)

8. Use the scroll bar on the right side to see the entire window.

   Look at the Description Field 1 and Description Field 2. Here, we entered normal text. See Figure 165 on page 216.



Product/Item Reference Number: 1449

Parent Reference Number:

Publish: Yes

Last Updated: 1999-04-11 18:04:1

Name/Short Description: ThinkPad 360PE

Description Field 1:

```
A full-function, lightweight, long-running
notebook. A quality product as all pruducts
produced by IBM.
```

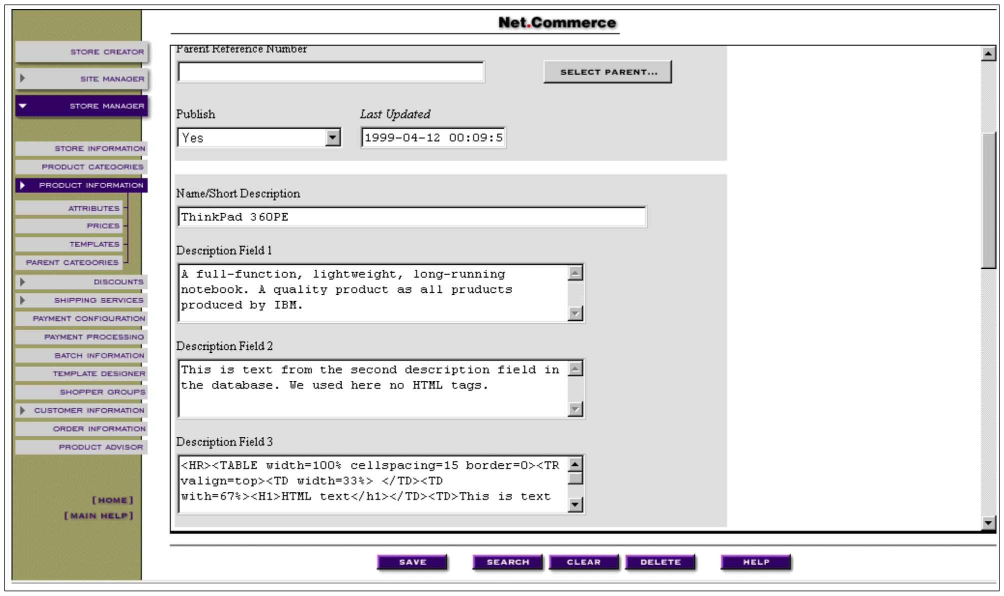
Description Field 2:

```
This is text from the second description field in
the database. We used here no HTML tags.
```

Description Field 3

Figure 165. Product/Item Information (Part 2 of 6)

9. In Description Field 3, we entered HTML coded text. See Figure 166.



Parent Reference Number:

Publish: Yes

Last Updated: 1999-04-12 00:09:5

Name/Short Description: ThinkPad 360PE

Description Field 1:

```
A full-function, lightweight, long-running
notebook. A quality product as all pruducts
produced by IBM.
```

Description Field 2:

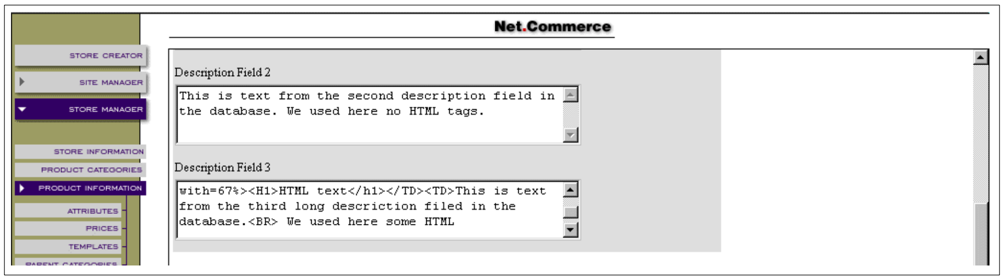
```
This is text from the second description field in
the database. We used here no HTML tags.
```

Description Field 3:

```
<HR><TABLE width=100% cellspacing=15 border=0><TR
valign=top><TD width=33%> </TD><TD
with=67%><H1>HTML text</h1></TD><TD>This is text
```

Figure 166. Product/Item Information (Part 3 of 6)

10. Use the scroll bar on the right side of the Description Field 3 to see more text (Figure 167 on page 217).



Description Field 2

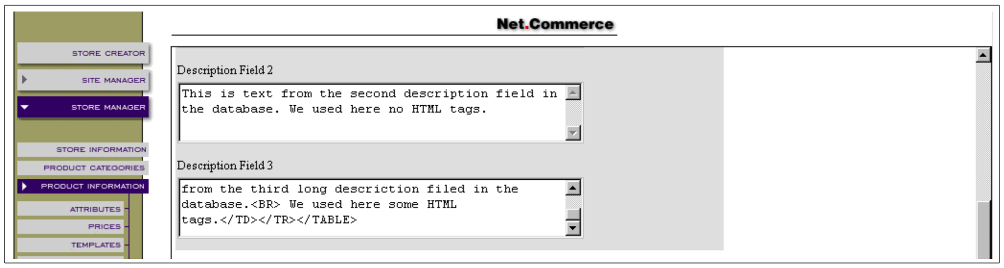
```
This is text from the second description field in
the database. We used here no HTML tags.
```

Description Field 3

```
with=67%><H1>HTML text</h1></TD><TD>This is text
from the third long descriction filed in the
database.<BR> We used here some HTML
```

Figure 167. Product/Item Information (Part 4 of 6)

11.Use the scroll bar on the right side of the Description Field 3 a second time to see more text (in Figure 168).



Description Field 2

```
This is text from the second description field in
the database. We used here no HTML tags.
```

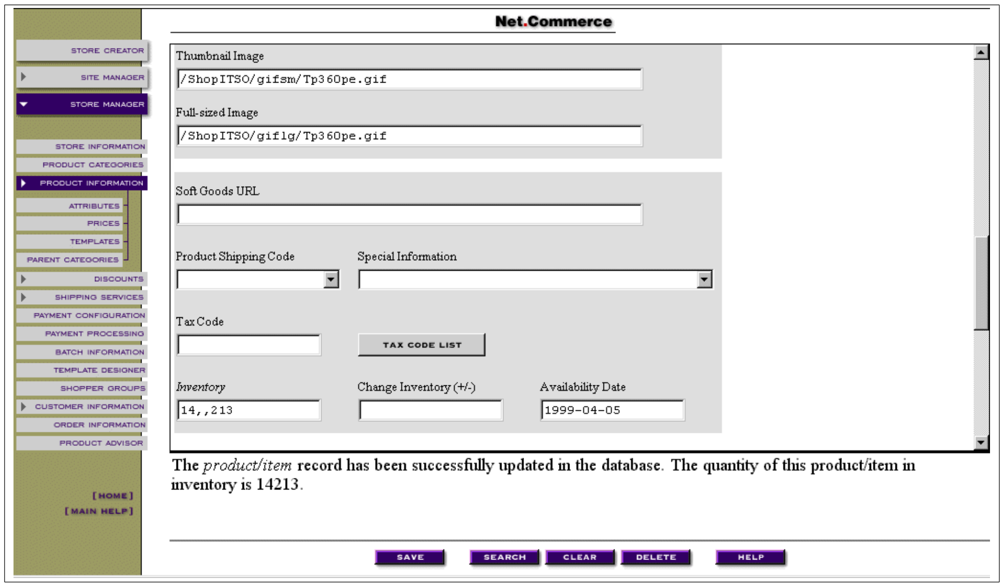
Description Field 3

```
from the third long descriction filed in the
database.<BR> We used here some HTML
tags.</TD></TR></TABLE>
```

Figure 168. Product/Item Information (Part 5 of 6)

12.Push the **SAVE** button.

13.If the product record is successfully updated, a message is displayed in the lower frame, as shown in Figure 169. Click the **RETURN** button to go back to the Product/Item page.



Thumbnail Image: /ShopITSO/gifsm/Tp360pe.gif

Full-sized Image: /ShopITSO/giflg/Tp360pe.gif

Soft Goods URL:

Product Shipping Code:

Special Information:

Tax Code:

Inventory: 14,,213

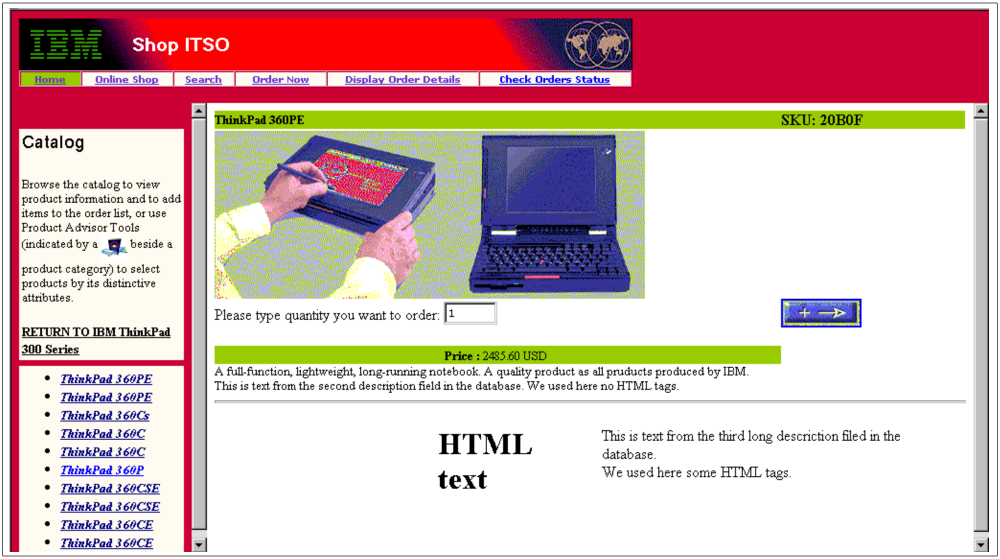
Change Inventory (+/-):

Availability Date: 1999-04-05

The *product/item* record has been successfully updated in the database. The quantity of this product/item in inventory is 14213.

Figure 169. Product/Item Information (Part 6 of 6)

Figure 170 on page 218 shows how the product page appears when you use HTML formatted text in the description field.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

ThinkPad 360PE SKU: 20B0F

Please type quantity you want to order: 1

Price : 2485.60 USD

A full-function, lightweight, long-running notebook. A quality product as all pruducts produced by IBM.
This is text from the second description field in the database. We used here no HTML tags.

HTML text

This is text from the third long descriction filed in the database.
We used here some HTML tags.

Figure 170. Product Page with HTML Formatted Long Description Field

13.11 Using the Product PRURL Field

In this section, we give you an example of how you can use the PRURL field in the PRODUCT table to make a link to an HTML file, which has additional description for the product. We did this only for one product. When you plan to use this approach for many or all of your products, you should use mass import to make your update in the product table for the PRURL field. See 3.2.3, “Planning Product Descriptions” on page 35.

We completed the following steps for one of our sample products:

1. Create an HTML source file with additional text. This can be any HTML source, perhaps with images or other multimedia files in it. See the following source code:

```
<HR><TABLE width=100% cellspacing=30 border=0>
<TR valign=top><TD width=33%> </TD>
<TD with=30%><H1>Additional Text</h1></TD>
<TD>This is text from the product.html that is assigned to the product
through the PRURL field in the PRODUCT table.<BR>
We used here some HTML tags.</TD></TR>
<A HREF="/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=1450&prmenbr=28"
TARGET="main">return to product page</A>
</TABLE>
<HR>
```

   Notice that we named our HTML source file based on the product SKU name. This is a good approach to have a logical name to know which source file belongs to which product. When you use mass import to fill the PRURL field of the product table, apply a meaningful name to a rule for matching the right values for the import file.

2. Update the PRODUCT table to include the value of the URL of the HTML source file. In our case, this file is stored in the same IFS directory where all our other HTML files for the store ShopITSO are stored. It is the directory /QIBM/UserData/NetCommerce/instance/work/html/ShopITSO/.

We used the following Pass statement in our HTTP Web Server Configuration file:

```
Pass /ShopITSO/*  /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/*
```

It maps the `/ShopITSO/` portion of the URL to the real directory where the files are located. See also A.11, “AS/400 Web Server Configuration File” on page 519. We can use the short name `/ShopITSO/20B0G.html` in the PRURL field of the PRODUCT table. See Figure 172 on page 220.

Because we activated the HTTP Web Server Local Cache for all the files that are stored in this directory, the 2oBoG.html HTML source file is also cached. See 13.14, “Using the HTTP Web Server Cache for Static Pages” on page 234.

Follow these steps to update the PRURL field throughout the Store Manager product forms:

a. Login as `ncadmin` in the Net.Commerce Administration Web page at the following URL:

   http://*fullyqualifiedhost_name*/ncadmin/

b. Select **Product Information** in the left frame of the window.

c. Select **Store ShopITSO** (your store name) in the Store selection list.

d. Choose **Product or Item** in the Input For selection list.

e. Click the **SEARCH** button.

Tip

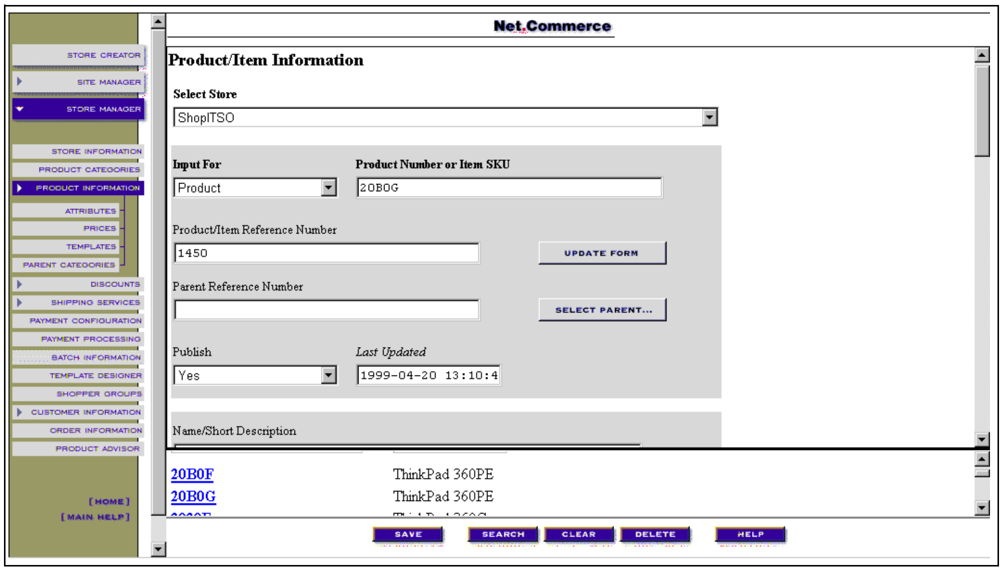
To get a larger second half of the screen, set the cursor of the black line and move this line above.

f. Click on the **product or item** that you want to change.

g. The chosen product or item is shown in the Product Number or Item SKU field, and you can see also all fields that you can update. See Figure 171 on page 220.

Tip

Make the second part of the window smaller to see the fields you can change.



Product/Item Information

Select Store: ShopITSO

Input For: Product

Product Number or Item SKU: 20B0G

Product/Item Reference Number: 1450

Parent Reference Number:

Publish: Yes

Last Updated: 1999-04-20 13:10:4

Name/Short Description

20B0F	ThinkPad 360PE
20B0G	ThinkPad 360PE

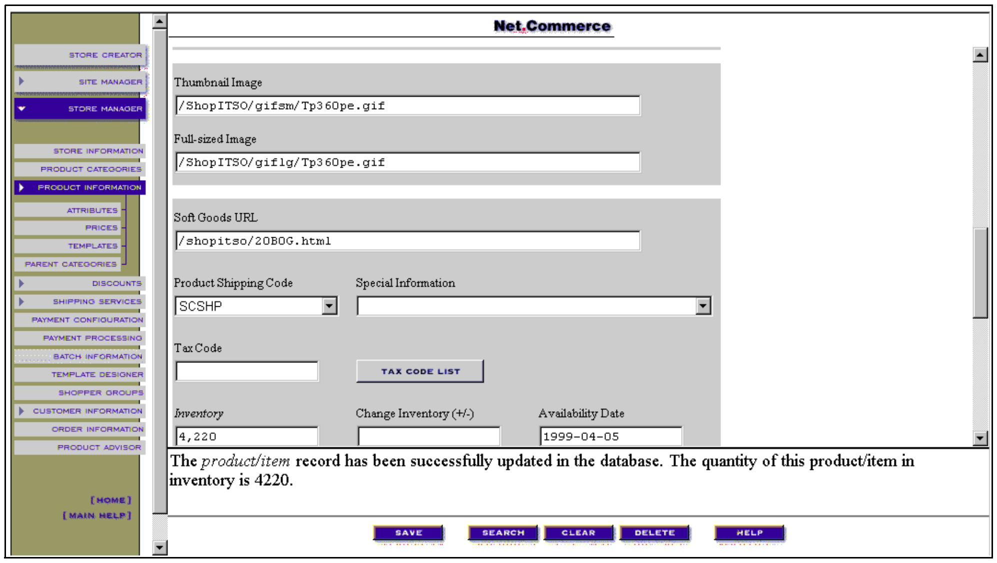
Figure 171. Update Product Table — PRURL Field (Part 1 of 2)

Use the scroll bar on the right side to see the entire window.

h. Look for the Soft Goods URL input field. Enter the value for the PRURL field here. See Figure 172.

i. Push the **SAVE** button.

j. If the product record is successfully updated, a message is displayed in the lower frame, as shown in Figure 172. Click the **RETURN** button to go back to the Product/Item page.



Thumbnail Image: /ShopITSO/gifsm/Tp360pe.gif

Full-sized Image: /ShopITSO/giflg/Tp360pe.gif

Soft Goods URL: /shopitso/20B0G.html

Product Shipping Code: SCSHP

Special Information:

Tax Code:

Inventory: 4,220

Change Inventory (+/-):

Availability Date: 1999-04-05

The *product/item* record has been successfully updated in the database. The quantity of this product/item in inventory is 4220.

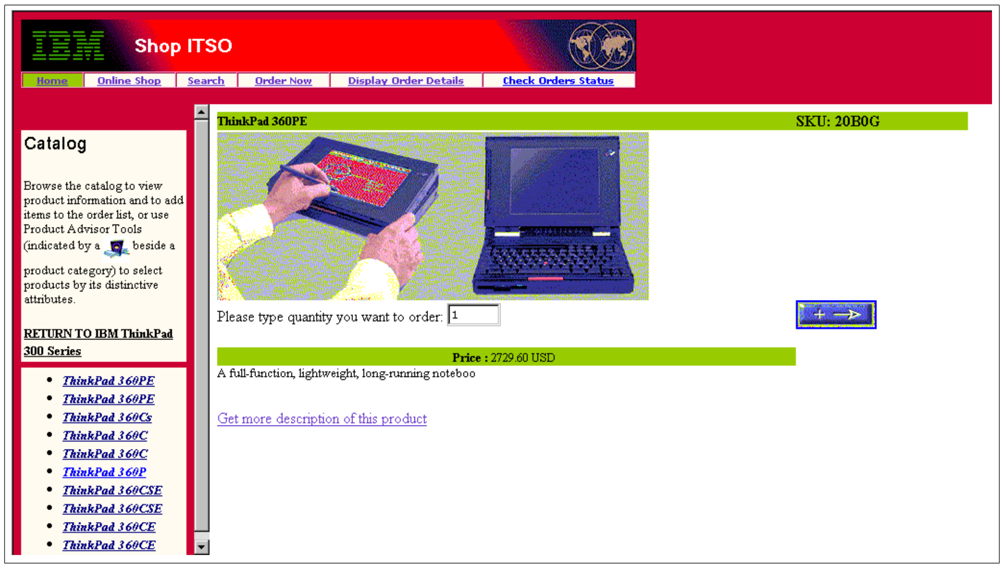
Figure 172. Update Product Table — PRURL Field (Part 2 of 2)

3. Create a new product template in which you also select the PRURL field in the SQL query and make the changes to display the link to this URL.

We did this on the basis of prod1.d2w. That means that we copied it, named it prod2.d2w, and made the previously mentioned changes. You can find this macro in A.9.4, "Product Macro PROD2.D2W" on page 487.

4. Assign the new product template to the product. To learn how to do this, see 13.8.2, "Assigning Templates to Products" on page 207.

Figure 173 shows the product page for a product that has the PRURL field filled in and uses the prod2.d2w template with the link to the description HTML file in the IFS.



Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

ThinkPad 360PE SKU: 20B0G

Please type quantity you want to order: 1

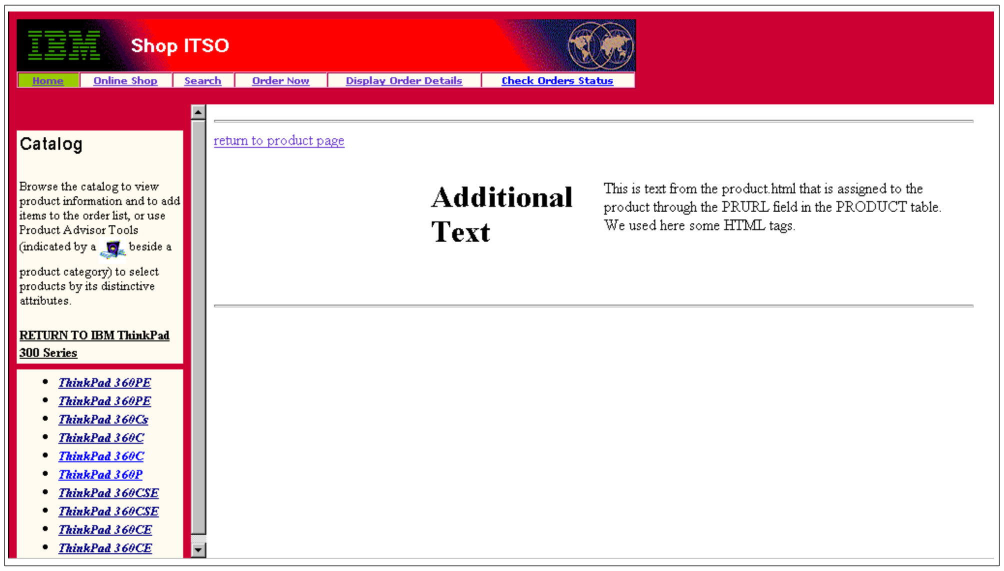
Price : 2729.60 USD

A full-function, lightweight, long-running noteboo

Get more description of this product

Figure 173. Product Page with Link to Description

Through the link to the 20B0g.html file, the customer can see the additional product description. See Figure 174. They can return from there to the Product page.



Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 300 Series

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

return to product page

Additional Text

This is text from the product.html that is assigned to the product through the PRURL field in the PRODUCT table. We used here some HTML tags.

Figure 174. Link Page from Product Page

13.12 Customizing the Category Tree

Because our back-end system delivers only a one-level category, the Top Category for all our products, we manually modified the category tree from the Net.Commerce Store Manager interface to add more categories. We also modified it to get a three-level category tree as a result.

After the mass import of our products, we obtained the one-level deep (flat) category tree shown in Figure 175.

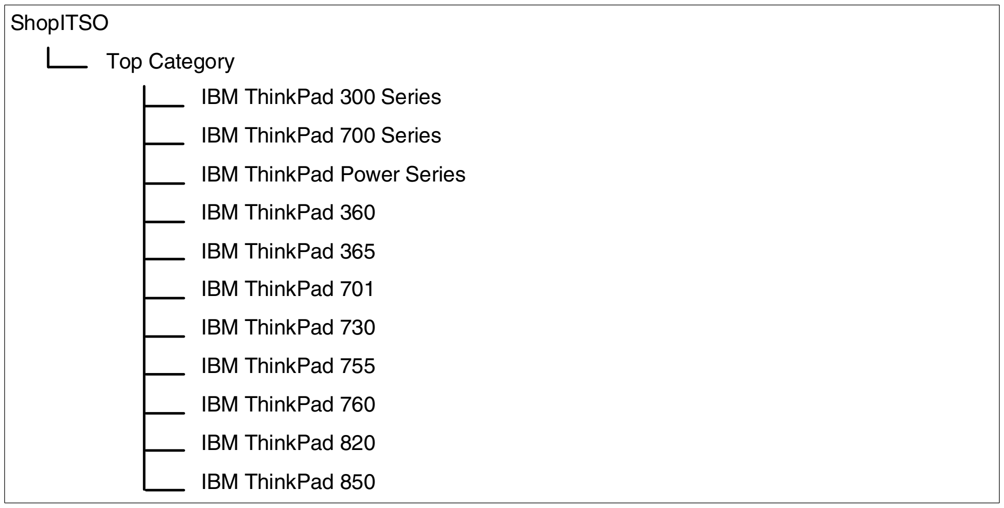


Figure 175. Category Tree — One Level Deep (Flat)

As shown in Figure 176 on page 223, we added three new one-level categories:

- IBM ThinkPads
- IBM Personal Computers
- IBM Servers

After creating the above mentioned one-level categories, we moved all our products from the Top Category to the IBM ThinkPads category. Notice that IBM ThinkPads is the only category with products. The other two categories are empty and were added only as examples.

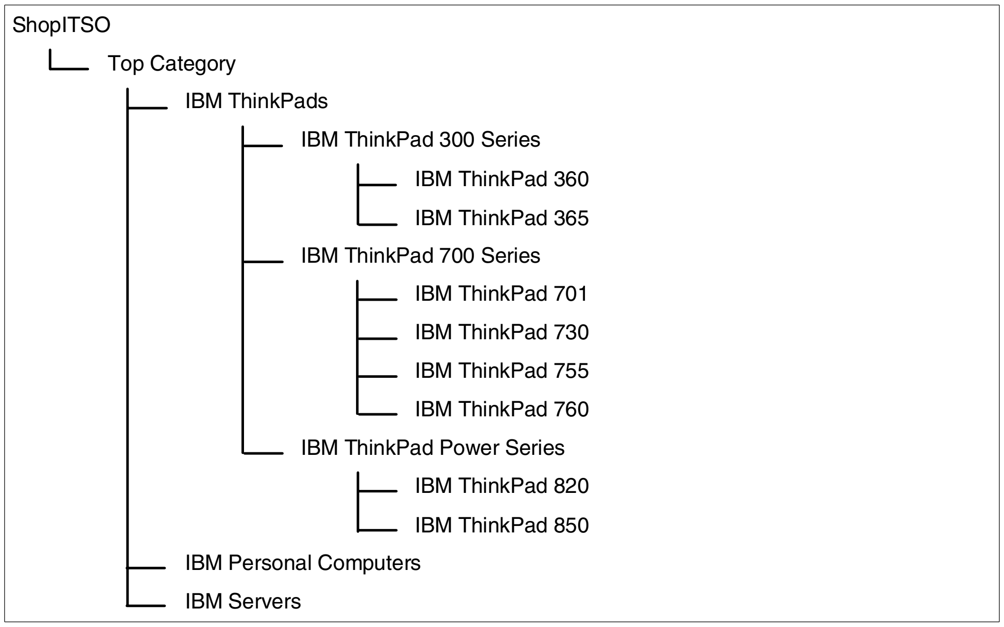


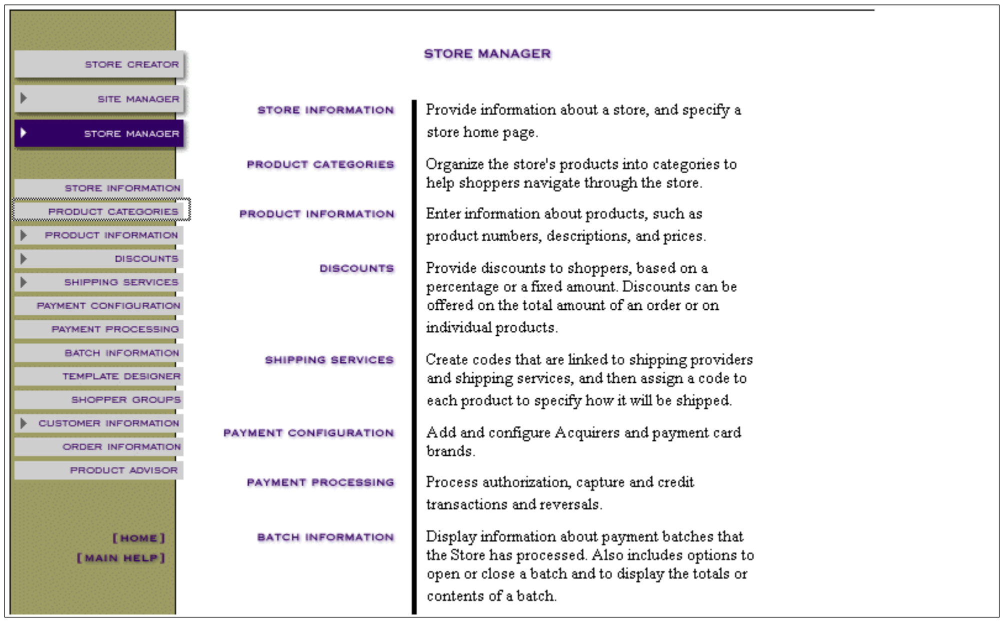
Figure 176. Category Tree — Three Levels

The following steps show how we did this customizing:

1. Log on as `ncadmin` in the Net.Commerce Administrator Web page at the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

2. Load the Product Categories page from the Store Manager view, as shown in Figure 177, by clicking the **Product Categories** button.

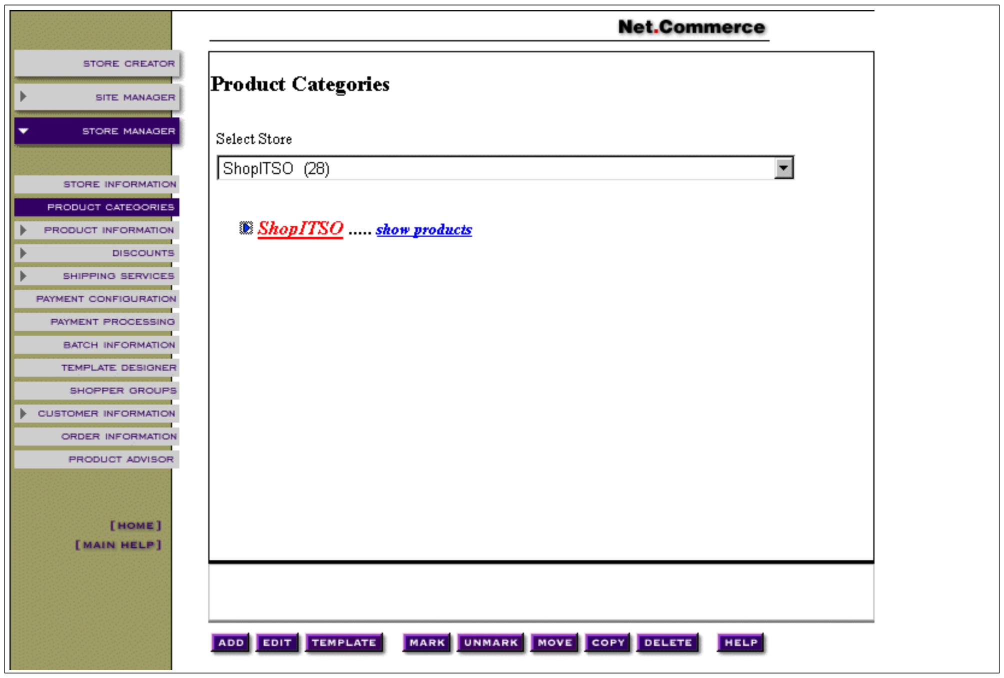


STORE MANAGER

STORE INFORMATION	Provide information about a store, and specify a store home page.
PRODUCT CATEGORIES	Organize the store's products into categories to help shoppers navigate through the store.
PRODUCT INFORMATION	Enter information about products, such as product numbers, descriptions, and prices.
DISCOUNTS	Provide discounts to shoppers, based on a percentage or a fixed amount. Discounts can be offered on the total amount of an order or on individual products.
SHIPPING SERVICES	Create codes that are linked to shipping providers and shipping services, and then assign a code to each product to specify how it will be shipped.
PAYMENT CONFIGURATION	Add and configure Acquirers and payment card brands.
PAYMENT PROCESSING	Process authorization, capture and credit transactions and reversals.
BATCH INFORMATION	Display information about payment batches that the Store has processed. Also includes options to open or close a batch and to display the totals or contents of a batch.

Figure 177. Store Manager View in Net.Commerce Administrator Page

3. Select the **ShopITSO** store and click the expand categories icon, as shown in Figure 178 on page 224, to show the root category of the store.



Product Categories

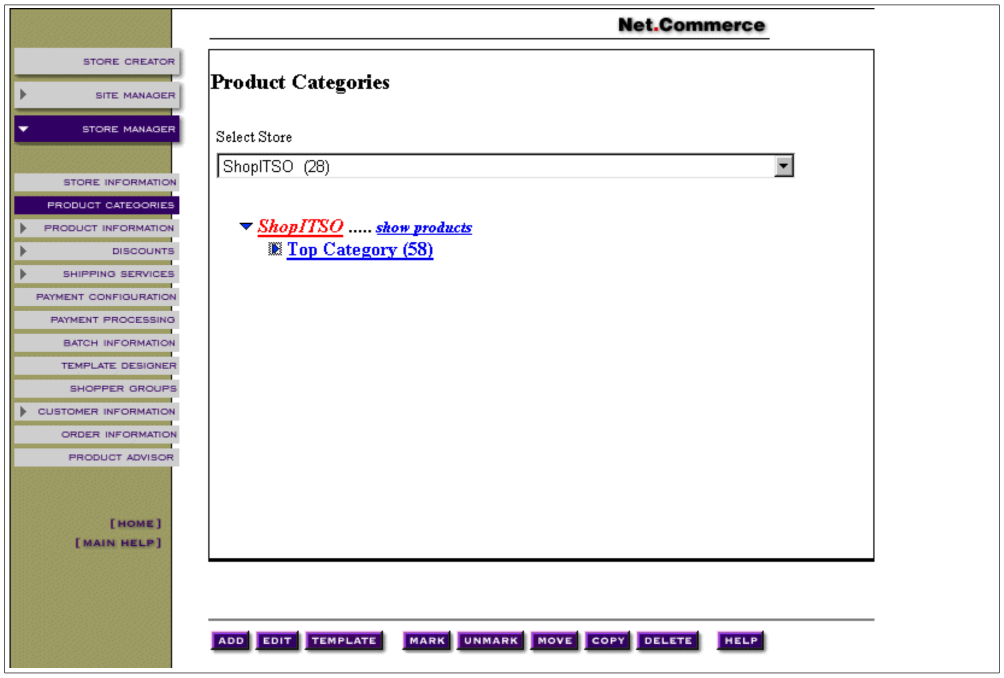
Select Store

ShopITSO (28)

ShopITSO ..... show products

Figure 178. Product Categories — ShopITSO Store

4. For the Top Category category, click the expand categories icon to show the level one categories (Figure 179).



Product Categories

Select Store

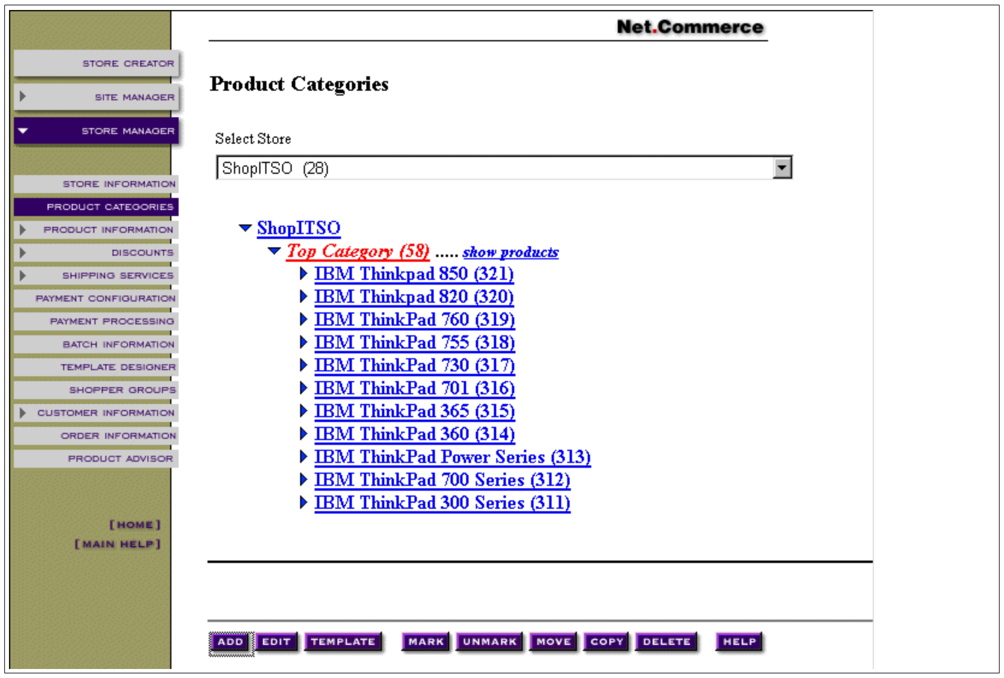
ShopITSO (28)

ShopITSO ..... show products

Top Category (58)

Figure 179. Product Categories — Top Category

5. Select Top Category and click the **Add** button, as shown in Figure 180 on page 225.



Product Categories

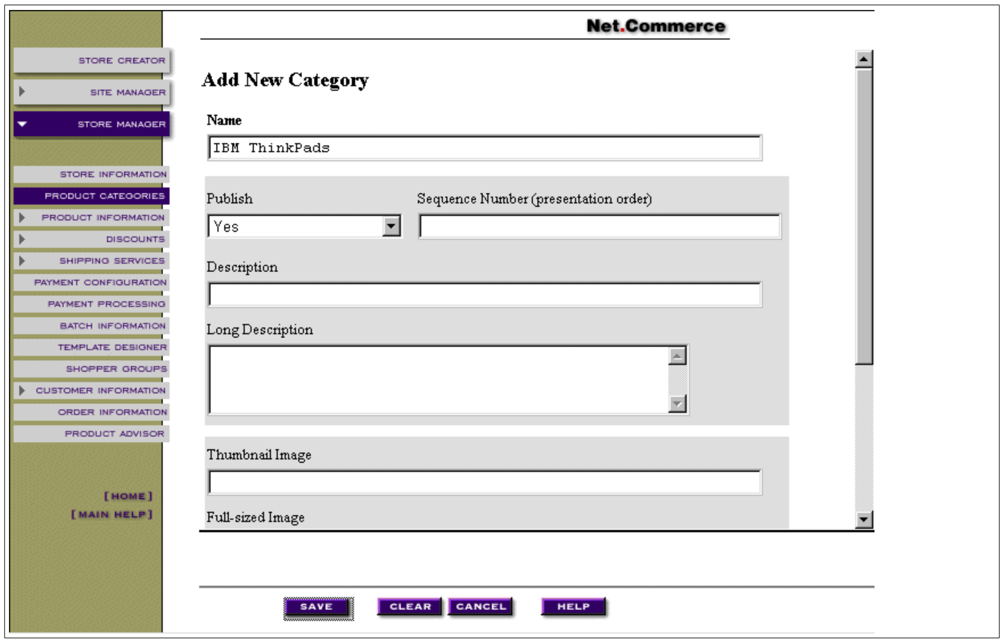
Select Store

ShopITSO (28)

- ShopITSO
  - Top Category (58) ..... show products
    - IBM Thinkpad 850 (321)
    - IBM Thinkpad 820 (320)
    - IBM ThinkPad 760 (319)
    - IBM ThinkPad 755 (318)
    - IBM ThinkPad 730 (317)
    - IBM ThinkPad 701 (316)
    - IBM ThinkPad 365 (315)
    - IBM ThinkPad 360 (314)
    - IBM ThinkPad Power Series (313)
    - IBM ThinkPad 700 Series (312)
    - IBM ThinkPad 300 Series (311)

Figure 180. Product Categories — Level One Categories (Select to Add Category)

6. Enter `IBM ThinkPads` as the new category name and click the **Save** button, as shown in Figure 181. You return to the Product Categories page.



Add New Category

Name: IBM ThinkPads

Publish: Yes

Sequence Number (presentation order):

Description:

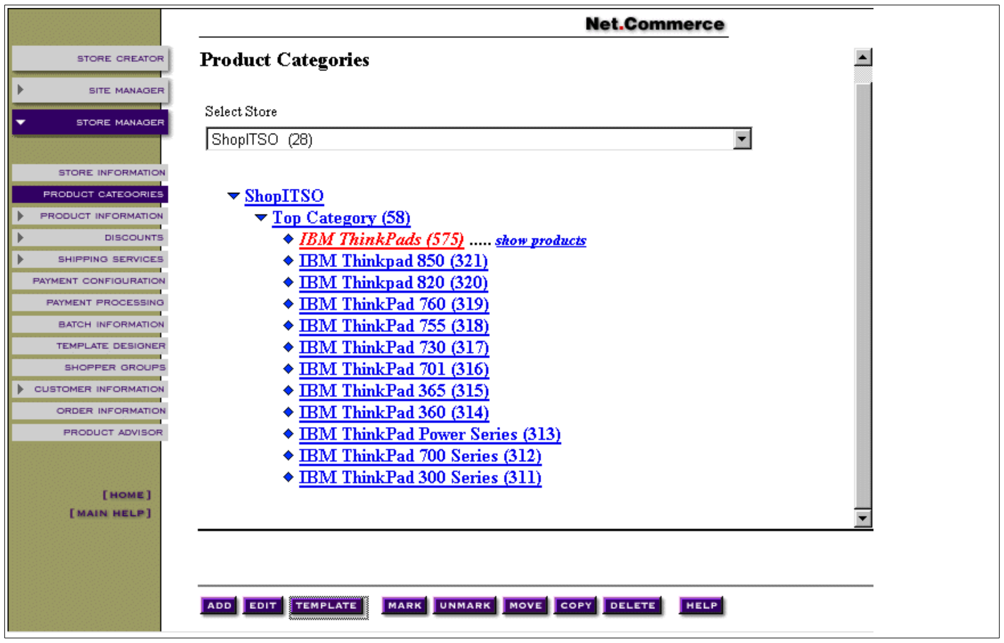
Long Description:

Thumbnail Image:

Full-sized Image

Figure 181. Products Categories — Add New Category

7. Select **IBM ThinkPads** and click the **Template** button to assign a category template (Figure 182 on page 226).



Product Categories

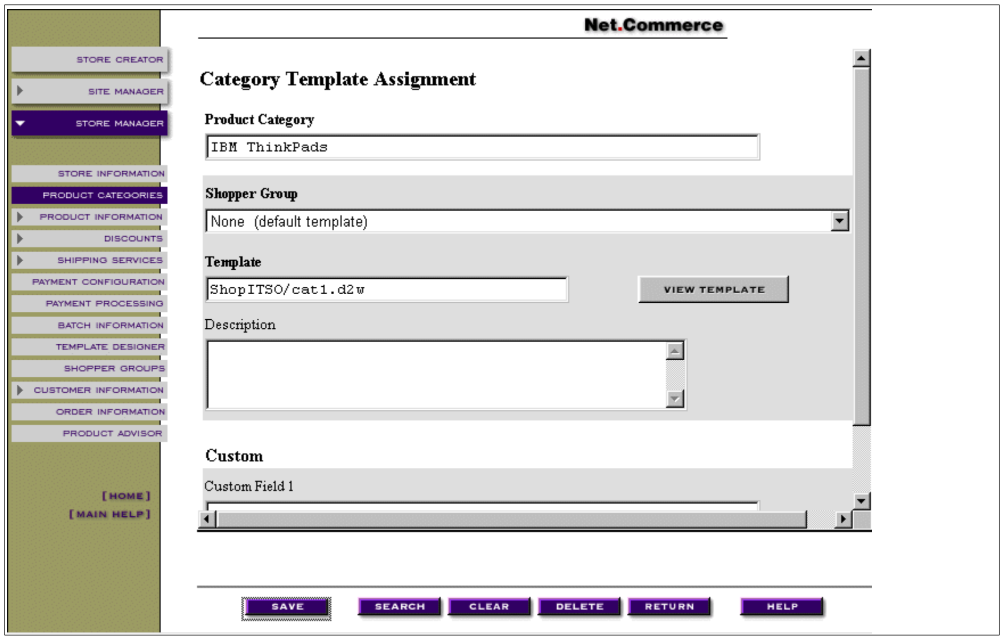
Select Store

ShopITSO (28)

- ShopITSO
  - Top Category (58)
    - IBM ThinkPads (575) ..... show products
    - IBM Thinkpad 850 (321)
    - IBM Thinkpad 820 (320)
    - IBM ThinkPad 760 (319)
    - IBM ThinkPad 755 (318)
    - IBM ThinkPad 730 (317)
    - IBM ThinkPad 701 (316)
    - IBM ThinkPad 365 (315)
    - IBM ThinkPad 360 (314)
    - IBM ThinkPad Power Series (313)
    - IBM ThinkPad 700 Series (312)
    - IBM ThinkPad 300 Series (311)

Figure 182. Products Categories — Level One Categories (Select to Assign Template)

8. Select **None (default template)** for the shopper group and select **ShopITSO/cat1.d2w** for the template. Click the **Save** button, as shown in Figure 183, to save the category template assignment.



Category Template Assignment

Product Category: IBM ThinkPads

Shopper Group: None (default template)

Template: ShopITSO/cat1.d2w

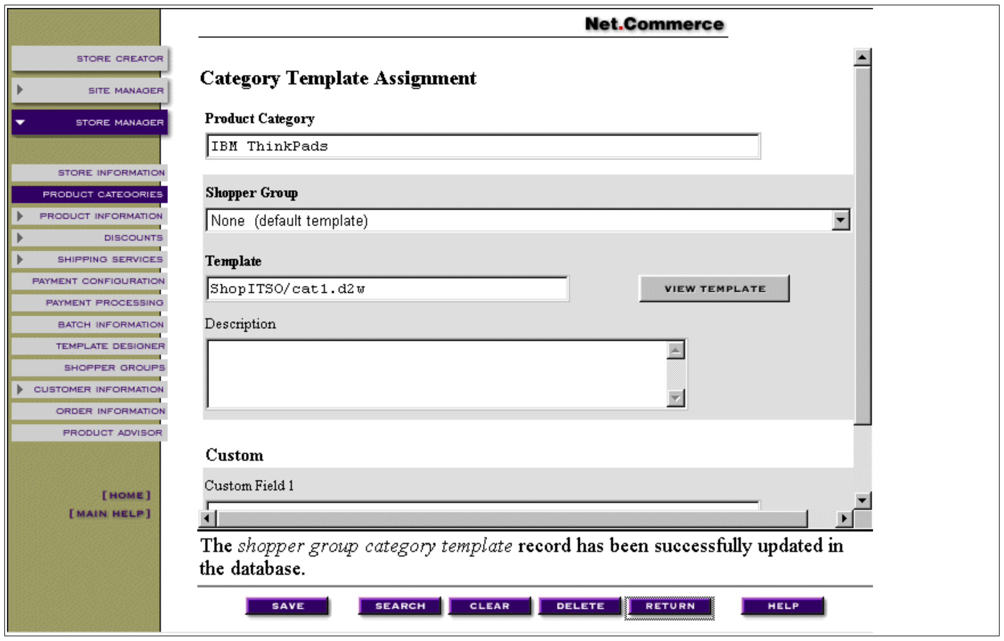
Description

Custom

Custom Field 1

Figure 183. Category Template Assignment — Save Template Assignment

9. If the category template record is successfully updated, a message is displayed in the lower frame, as shown in Figure 184 on page 227. Click **RETURN** to go back to the Product Categories page.



Category Template Assignment

Product Category: IBM ThinkPads

Shopper Group: None (default template)

Template: ShopITSO/cat1.d2w

Description

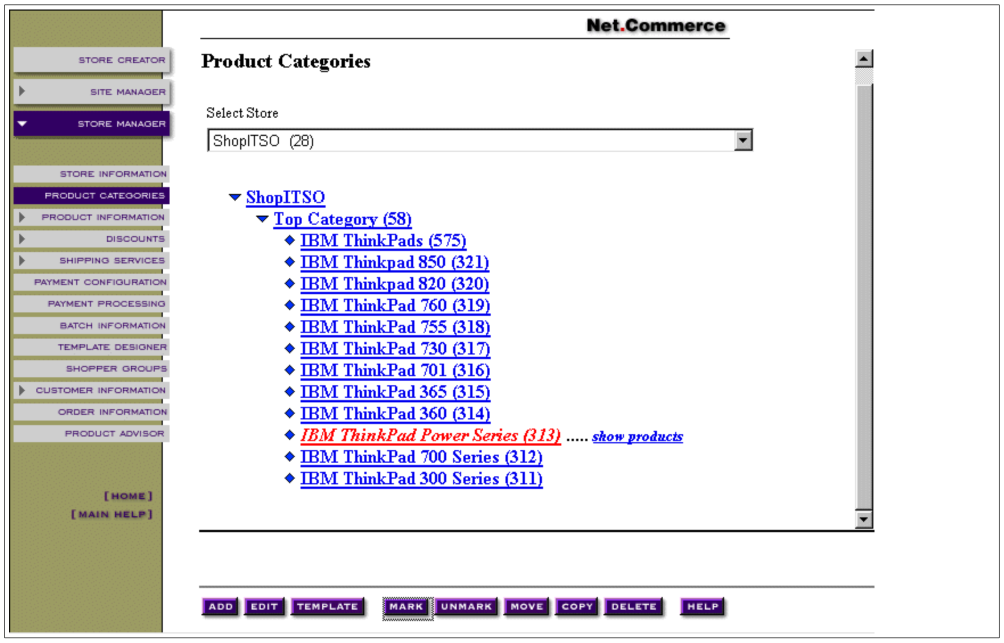
Custom

Custom Field 1

The *shopper group category template* record has been successfully updated in the database.

Figure 184. Category Template Assignment — Return to Product Categories

10.To move the IBM ThinkPad Power Series from category level one to category level two under IBM ThinkPads, click to select the IBM ThinkPad Power Series category. Then, click the **MARK** button as shown in Figure 185.



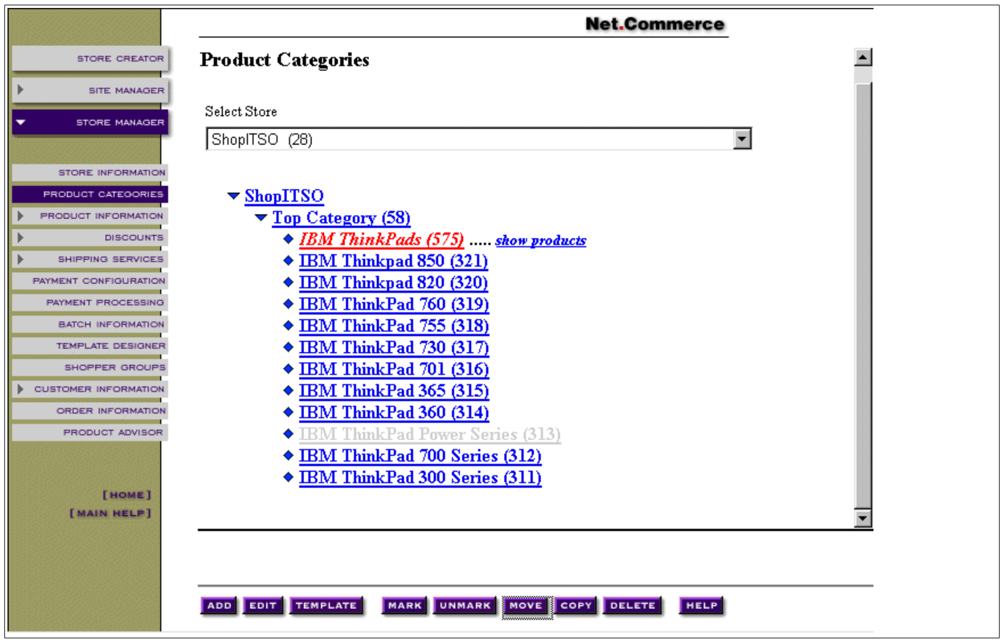
Product Categories

Select Store: ShopITSO (28)

- ShopITSO
  - Top Category (58)
    - IBM ThinkPads (575)
    - IBM Thinkpad 850 (321)
    - IBM Thinkpad 820 (320)
    - IBM ThinkPad 760 (319)
    - IBM ThinkPad 755 (318)
    - IBM ThinkPad 730 (317)
    - IBM ThinkPad 701 (316)
    - IBM ThinkPad 365 (315)
    - IBM ThinkPad 360 (314)
    - *IBM ThinkPad Power Series (313)* ..... *show products*
    - IBM ThinkPad 700 Series (312)
    - IBM ThinkPad 300 Series (311)

Figure 185. Products Categories — Level One Categories (Select to Mark)

11.Click on the IBM ThinkPads category you need to select it. Click the **MOVE** button as shown in Figure 186 on page 228.



Product Categories

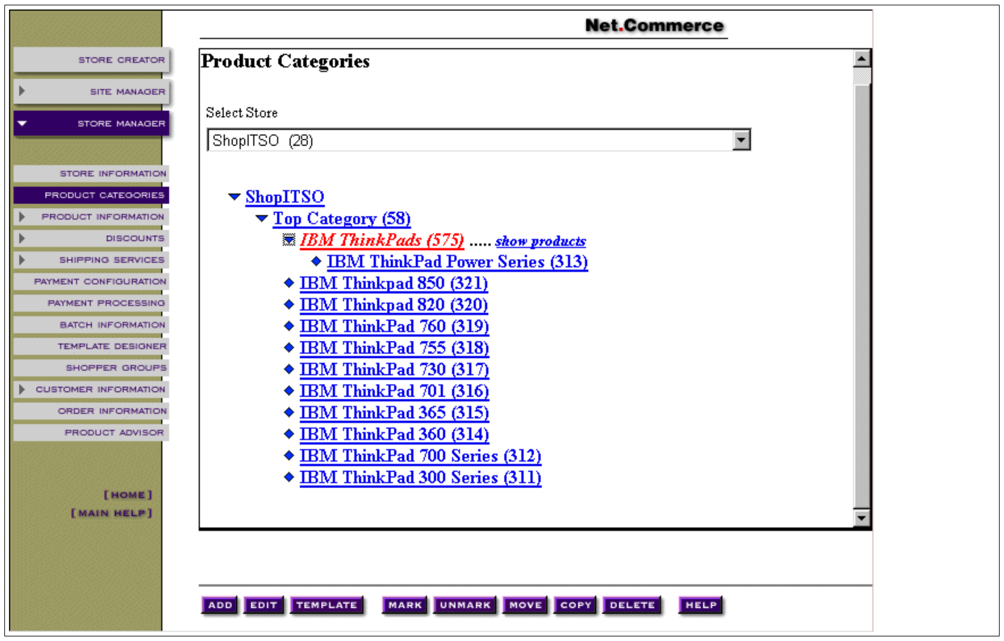
Select Store

ShopITSO (28)

- ShopITSO
  - Top Category (58)
    - IBM ThinkPads (575) ..... show products
    - IBM Thinkpad 850 (321)
    - IBM Thinkpad 820 (320)
    - IBM ThinkPad 760 (319)
    - IBM ThinkPad 755 (318)
    - IBM ThinkPad 730 (317)
    - IBM ThinkPad 701 (316)
    - IBM ThinkPad 365 (315)
    - IBM ThinkPad 360 (314)
    - IBM ThinkPad Power Series (313)
    - IBM ThinkPad 700 Series (312)
    - IBM ThinkPad 300 Series (311)

Figure 186. Products Categories — Level One Categories (Select to Move)

12.After the category is moved, click the expand category icon, as shown in Figure 187, to display category levels one and two of the modified category tree.



Product Categories

Select Store

ShopITSO (28)

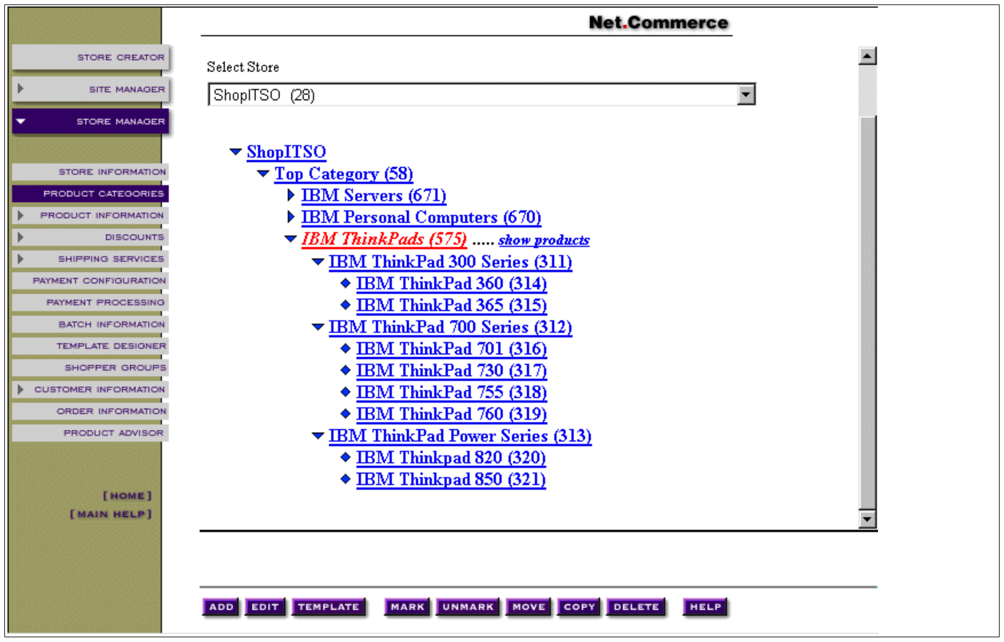
- ShopITSO
  - Top Category (58)
    - IBM ThinkPads (575) ..... show products
      - IBM ThinkPad Power Series (313)
    - IBM Thinkpad 850 (321)
    - IBM Thinkpad 820 (320)
    - IBM ThinkPad 760 (319)
    - IBM ThinkPad 755 (318)
    - IBM ThinkPad 730 (317)
    - IBM ThinkPad 701 (316)
    - IBM ThinkPad 365 (315)
    - IBM ThinkPad 360 (314)
    - IBM ThinkPad 700 Series (312)
    - IBM ThinkPad 300 Series (311)

Figure 187. Products Categories — Level One and Two Categories

13.Repeat steps 5 through 9 to create the other two one-level categories: IBM Personal Computers and IBM Servers.

14.Repeat steps 10 through 12 to move the rest of the categories under IBM ThinkPads to create level two and three categories. Refer to Figure 176 on page 223.

The final three-level category tree is shown in Figure 188.



- ShopITSO
  - Top Category (58)
    - IBM Servers (671)
    - IBM Personal Computers (670)
    - IBM ThinkPads (575) ..... show products
      - IBM ThinkPad 300 Series (311)
        - IBM ThinkPad 360 (314)
        - IBM ThinkPad 365 (315)
      - IBM ThinkPad 700 Series (312)
        - IBM ThinkPad 701 (316)
        - IBM ThinkPad 730 (317)
        - IBM ThinkPad 755 (318)
        - IBM ThinkPad 760 (319)
      - IBM ThinkPad Power Series (313)
        - IBM Thinkpad 820 (320)
        - IBM Thinkpad 850 (321)

Figure 188. Products Categories — Level One, Two, and Three Categories

13.13 Customizing the HTML Pages

Because of performance considerations, we decided to use HTML pages for all the static pages and frames of the site. Only the dynamically generated pages are served by Net.Commerce in our shop.

During the configuration of the Net.Commerce instance, we enabled the caching functionality for our Net.Commerce instance to serve dynamically generated pages from the cache directory. We did this instead of accessing the database for pages that were already served.

Keep in mind that only pages, which are called by the Net.Commerce commands CategoryDisplay and ProductDisplay, are cached. Pages that are generated by Net.Data macro (for example, Display Current Order). Pages that are called through the ExecMacro Net.Commerce command are not cached.

The HTML files, Net.Data macros, and the Net.Data include files that are generated by the Store Creator are listed in Table 13 on page 230. This table also shows if the resulting Web page is dynamically generated by an SQL query to the database, and if it can be cached by the Net.Commerce server.

Also notice that the Web pages, such as the banner, frames, main, and navigation side, are not generated dynamically using SQL queries. Their content is pure HTML.

Finally notice that the home page is a dynamically generated page that uses SQL queries in the Net.Data macro.

Table 13. Items Generated by Store Creator

IFS Path	Dynamic	Cached
instance_path/html/store_name/index.html	N/A	No
instance_path/macro/category/*store_name*/cat1.d2w	Yes	**Yes**
instance_path/macro/product/*store_name*/prod1.d2w	Yes	**Yes**
instance_path/macro/store_name/banner.d2w	**No**	**No**
instance_path/macro/store_name/contact.d2w	Yes	No
instance_path/macro/*store_name*/err_adrbk_up.d2w	Yes	No
instance_path/macro/*store_name*/err_check_inv.d2w	Yes	No
instance_path/macro/*store_name*/err_shaddr.d2w	Yes	No
instance_path/macro/tore_name/frames.d2w	**No**	**No**
instance_path/macro/store_name/home.d2w	**Yes**	**No**
instance_path/macro/store_name/main.d2w	**No**	**No**
instance_path/macro/tore_name/nav_side.d2w	**No**	**No**
instance_path/macro/*store_name*/order.d2w	Yes	No
instance_path/macro/*store_name*/oderlstc.d2w	Yes	No
instance_path/macro/*store_name*/orderok.d2w	Yes	No
instance_path/macro/*store_name*/ordnone.d2w	Yes	No
instance_path/macro/store_name/search.d2w	**Yes**	**No**
instance_path/macro/*store_name*/searchrslt.d2w	Yes	No
instance_path/macro/*store_name*/shipto.d2w	Yes	No
instance_path/macro/*store_name*/ShopITSO.inc	N/A	N/A

In Table 13, the *instance_path* is /QIBM/UserData/NetCommerce/instance/*instance_name*.

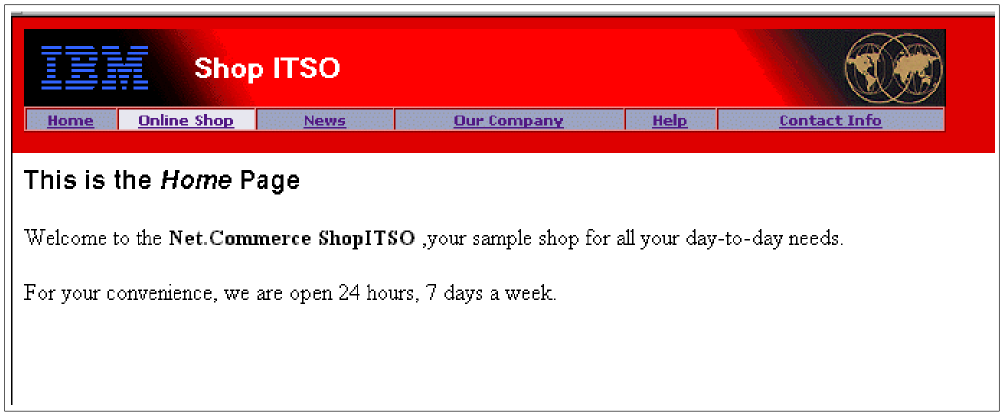
We do not use the marked Net.Data macros in Table 13 for our ShopITSO sample. For all of these macros, we build new HTML files that are cached by the AS/400 HTTP server using the Local cache function. See 13.14, “Using the HTTP Web Server Cache for Static Pages” on page 234. You can find our sources for the HTML pages in A.8, “HTML Samples” on page 475.

We strictly followed our navigation flow from the design. See 3.9.4, “Navigation Flow and Net.Commerce Commands in ShopITSO” on page 62.

The figures on the following pages show all of our HTML pages that we used in ShopITSO. Notice that our pages are very simple. In a real store, you do more work here to get fancy and informational pages.

The first page that the shopper sees is our home page. See Figure 189 on page 231. It is build by the Index.html file, which controls the frameset with two frames, the banner frame, which presents the banner1.html, and the main frame, which presents here our home page (home.html).

All HTML pages that you can call through banner1.html are displayed in the main frame. The Online Shop is an exception. See the explanation following Figure 193 on page 233.



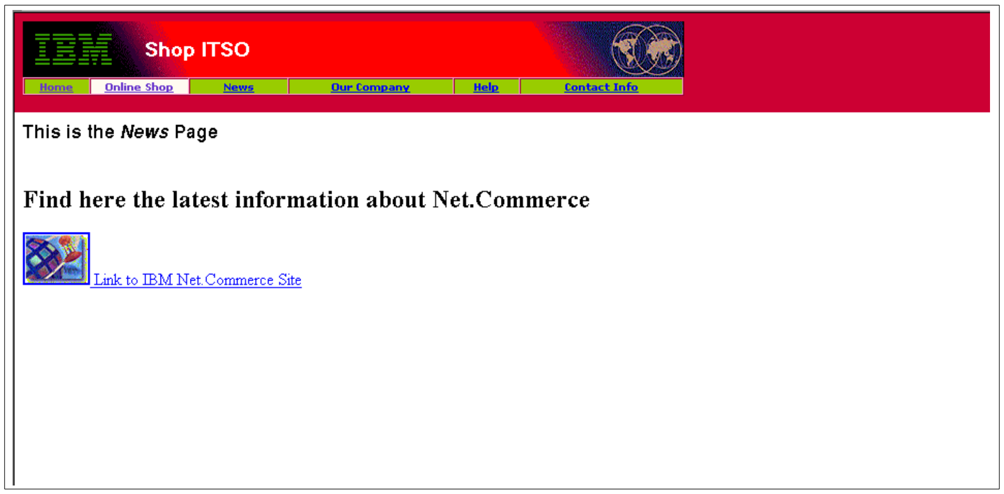
This is the *Home* Page

Welcome to the **Net.Commerce ShopITSO** ,your sample shop for all your day-to-day needs.

For your convenience, we are open 24 hours, 7 days a week.

Figure 189. ShopITSO Home Page with First Banner

The News Page (news.html) has a link to the IBM Net.Commerce Site where you can get the latest information about IBM Net.Commerce.



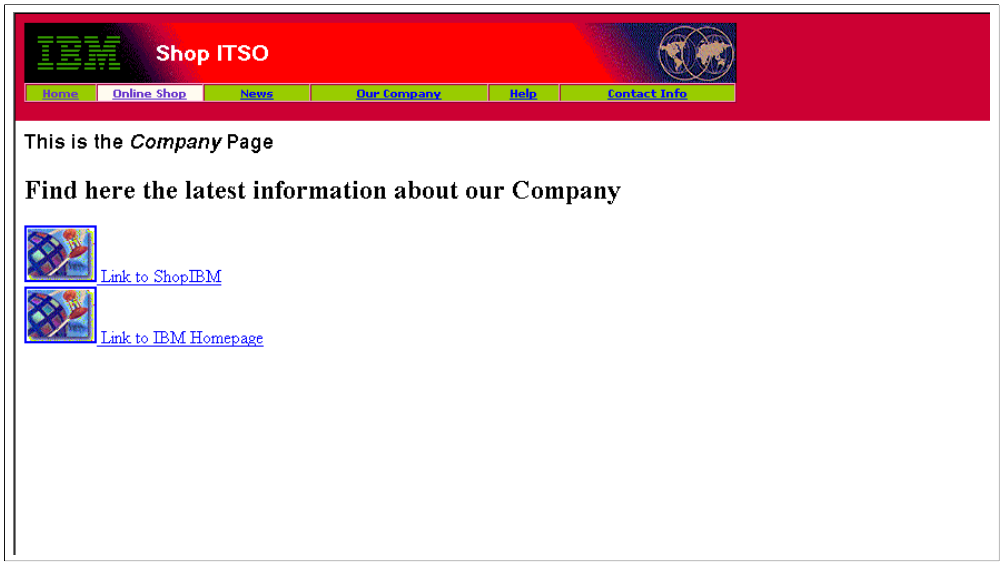
This is the *News* Page

Find here the latest information about Net.Commerce

Link to IBM Net.Commerce Site

Figure 190. ShopITSO News Page

Our Company (company.html) has two links to IBM sites on the World Wide Web. See Figure 191 on page 232.



This is the *Company* Page

Find here the latest information about our Company

Link to ShopIBM

Link to IBM Homepage

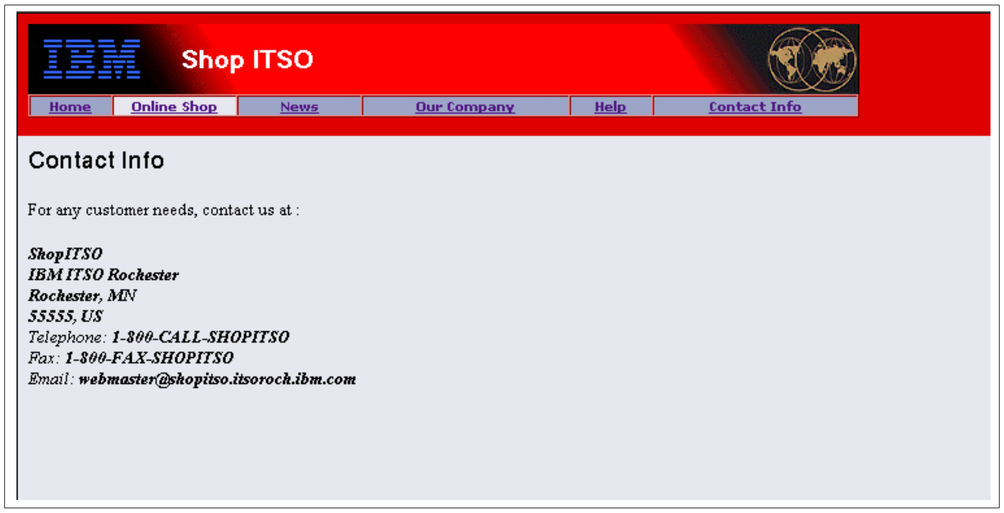
Figure 191. ShopITSO Our Company Page

The Help Page (help.html) is still under construction (Figure 192).

This is the ***Help*** Page

Figure 192. ShopITSO Help Page

The Contact Info page (contact.html) has the same information as the original contact page that is delivered with the Store Creator. The difference is that the original page was built by a Net.Data macro with an SQL query to get the information about the merchant from the merchant table. Because this information is more static (it changes not or very seldom), it can use an HTML page with static text.



Contact Info

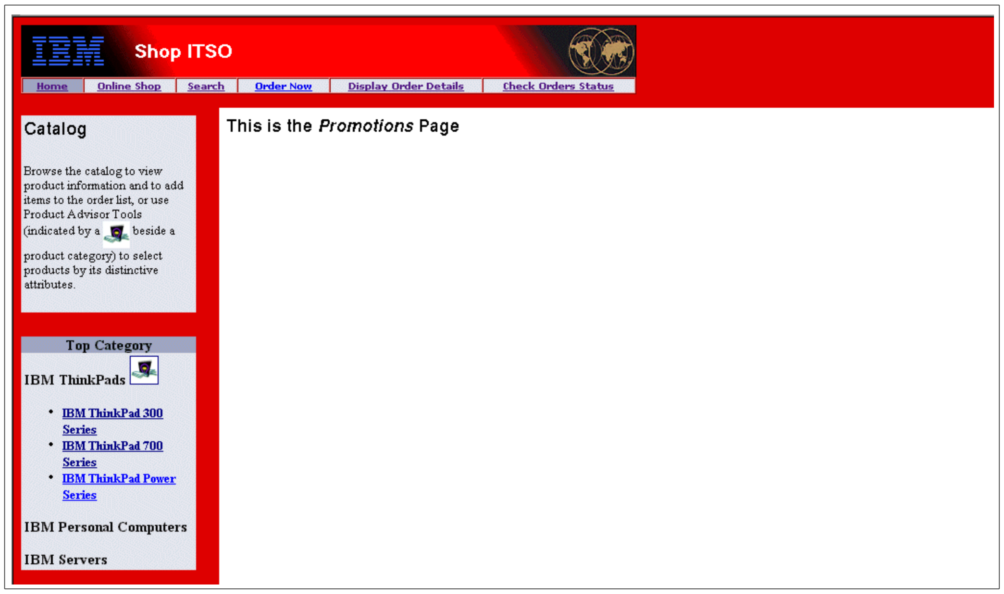
For any customer needs, contact us at :

ShopITSO
IBM ITSO Rochester
Rochester, MN
55555, US
Telephone: ***1-800-CALL-SHOPITSO***
Fax: ***1-800-FAX-SHOPITSO***
Email: ***webmaster@shopitso.itsoroch.ibm.com***

Figure 193. ShopITSO Contact Info Page

The link from the Online Shop button from banner1.html evokes the catalog.html, which starts our Net.Commerce e-business application.

It controls two framesets: the banner2.html, which presents the banner, and the navigation bars. The second is divided into two frames. Our catalog tree and the main frame, which presents our promotions page (promotions.html), is on the left side frame (named left). See Figure 194. For more information about our catalog tree, see 3.9.4.3, “Navigation Flow from Online Shop” on page 65.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

Top Category

IBM ThinkPads

- IBM ThinkPad 300 Series
- IBM ThinkPad 700 Series
- IBM ThinkPad Power Series

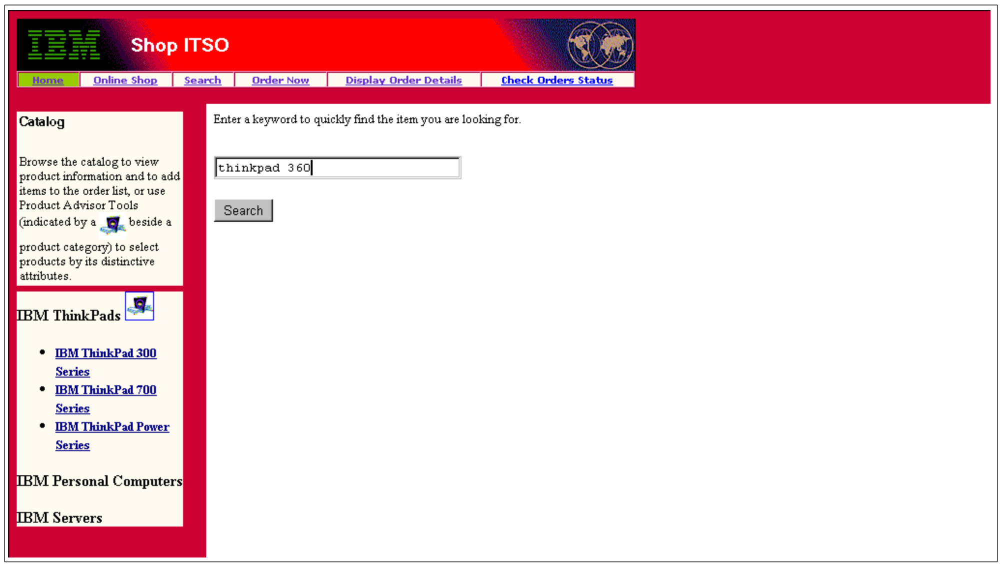
IBM Personal Computers

IBM Servers

This is the *Promotions* Page

Figure 194. ShopITSO Online Shop Page with Promotion Page

The Search page (search.html) is an HTML page where the customer can enter a search string to search for a product. For more information about our search function, see 3.9.4.10, “Navigation Flow Search Results Page” on page 80.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

IBM ThinkPads

- IBM ThinkPad 300 Series
- IBM ThinkPad 700 Series
- IBM ThinkPad Power Series

IBM Personal Computers

IBM Servers

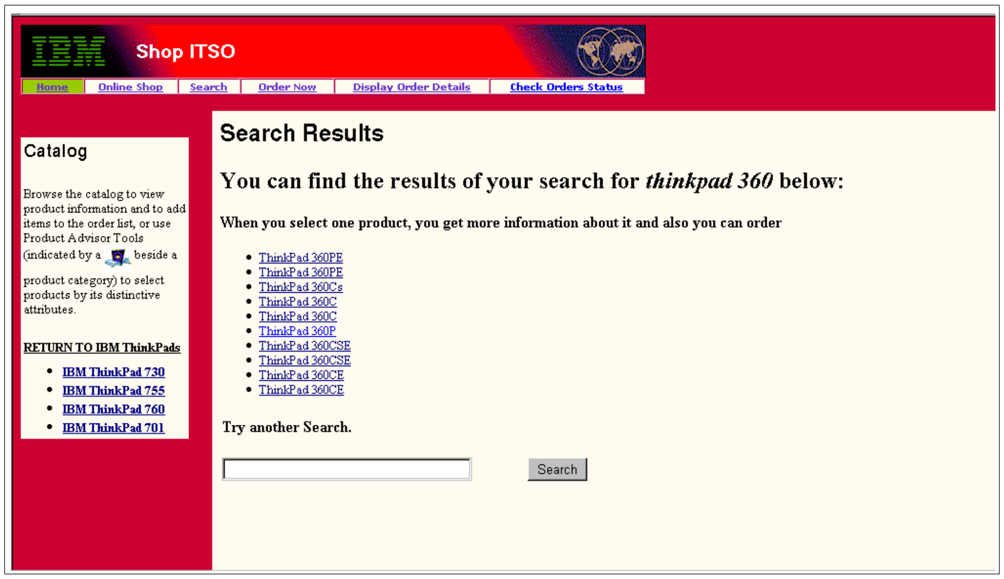
Enter a keyword to quickly find the item you are looking for.

thinkpad 360

Search

Figure 195. ShopITSP Search Page Part1

The search.html uses the NetCommerce command ExecMacro to invoke the Net.Data macro searchrst.d2w/report, which shows the Search Result page (Figure 196).



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPads

- IBM ThinkPad 730
- IBM ThinkPad 755
- IBM ThinkPad 760
- IBM ThinkPad 701

Search Results

You can find the results of your search for *thinkpad 360* below:

When you select one product, you get more information about it and also you can order

- ThinkPad 360PE
- ThinkPad 360PE
- ThinkPad 360Cs
- ThinkPad 360C
- ThinkPad 360C
- ThinkPad 360P
- ThinkPad 360CSE
- ThinkPad 360CSE
- ThinkPad 360CE
- ThinkPad 360CE

Try another Search.

Search

Figure 196. ShopITSO Search Result Page

The page for Order Now is built by a Net.Data macro. The pages Display Order Details and Check Order Status are invoked by Net.Commerce commands.

13.14 Using the HTTP Web Server Cache for Static Pages

By keeping your most frequently served files loaded in the server's memory, you can improve your server response time for those files. For example, consider that you load your server home page into memory at startup by adding it to the cache list. In this case, the server can handle requests for the page much more quickly than if it had to read the file from a disk. Keep in mind that for each file you load into memory, you are making that amount of memory unavailable for other uses that can affect performance.

In our sample ShopITSO store, we use the AS/400 HTTP Web Server Local caching function for all our static HTML pages such as the home page and all images. To activate the Local caching on the AS/400 HTTP Server, make the following entry in your HTTP server configuration file:

```
Maximum memory for file caching: Parameter name CacheLocalMaxBytes
```

Complete the following steps:

1. Define the maximum memory for file caching. Use a value from 1 KB through 32,767 MB.

   ```
   CacheLocalMaxBytes       2 M
   ```

2. Define the maximum number of files you want stored in the cache at one time in the Maximum number of files to cache.

   ```
   CacheLocalMaxFiles       200
   ```

3. Use LiveLocalCache On when a file is modified, and the server will serve the latest version of cached files. Otherwise, use Off.

   ```
   LiveLocalCache       Off
   ```

4. Make an entry for all the static files (.html and .gif files in our store) that you want to cache. You can use a generic name (*) for the file name for all files in a path, but you have to type the full path:

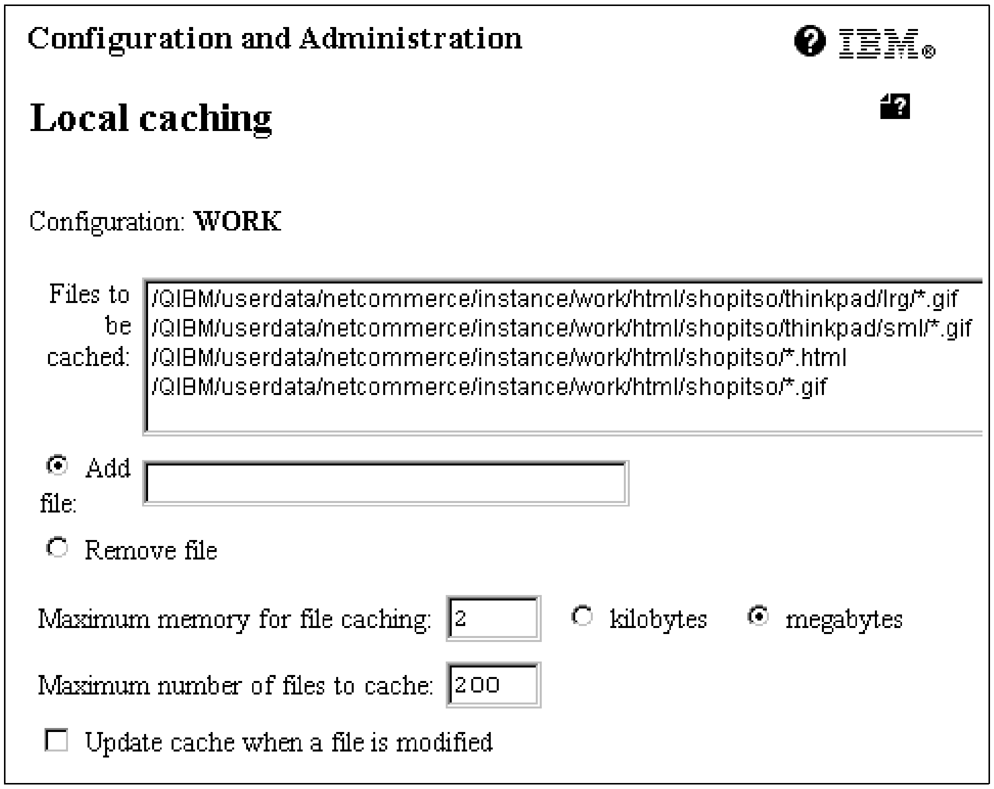
   ```
   CacheLocalFile
   /QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/lrg/*.gif

   CacheLocalFile
   /QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/sml/*.gif

   CacheLocalFile
   /QIBM/userdata/netcommerce/instance/work/html/shopitso/*.html

   CacheLocalFile /QIBM/userdata/netcommerce/instance/work/html/shopitso/*.gif
   ```

Note

Changes you make to the Local caching form, the Performance form, and the Timeouts form in the HTTP server configuration file all influence the performance of your server.

You can make your entry through the HTTP server administration and configuration Web page using the Local caching form. Or, use the AS/400 command `WRKHTTPCFG (<instance_name>)` to edit the HTTP configuration file.

The values we set in the HTTP server configuration file for our HTTP instance work are shown in Figure 197 on page 236.


Configuration and Administration

Local caching

Configuration: **WORK**

Files to be cached:
/QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/lrg/*.gif
/QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/sml/*.gif
/QIBM/userdata/netcommerce/instance/work/html/shopitso/*.html
/QIBM/userdata/netcommerce/instance/work/html/shopitso/*.gif

[✓ Add file:]

Remove file

Maximum memory for file caching: 2 kilobytes [✓ megabytes]

Maximum number of files to cache: 200

☐ Update cache when a file is modified

Figure 197. Configuration Form for Local Caching — HTTP Server Configuration

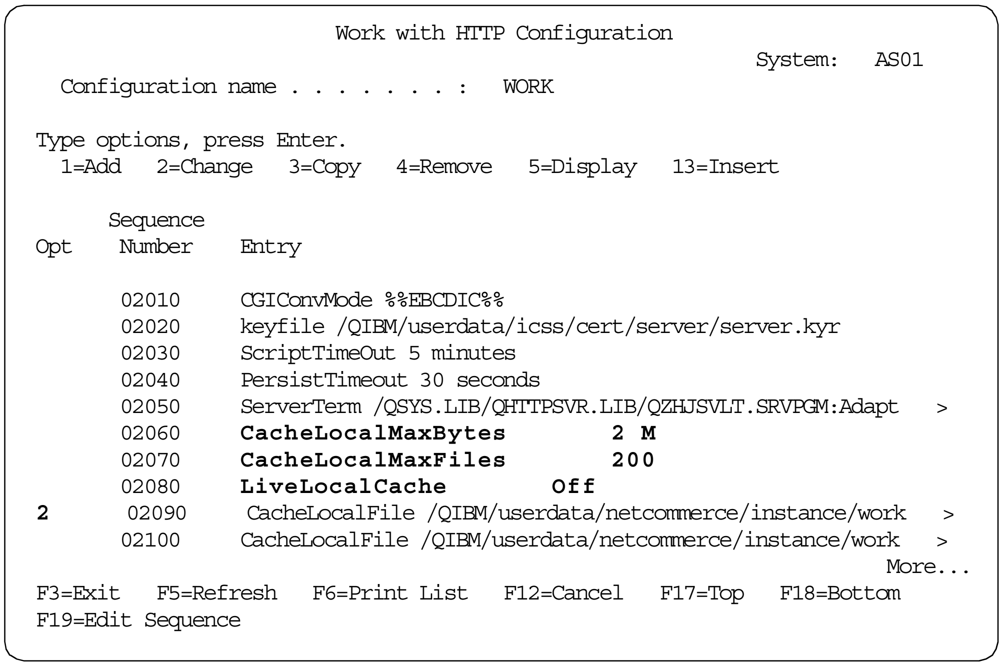
After you make the entries for the Local caching function, you have to end and restart the HTTP server instance. Do this with the following AS/400 commands:

```
ENDTCPSVR SERVER (*HTTP) HTTPSVR(<instance_name>)
STRTCPSVR SERVER (*HTTP) HTTPSVR(<instance_name>)
```

Note

We found that it was not possible to type in a path name and a generic file name with the * value in the Local caching form of the Web-based administration function. To avoid this problem, we used the AS/400 `WRKHTTPCFG` command to make these entries.

The following figures illustrate the screens that are shown when using the `WRKHTTPCFG` AS/400 command. Figure 198 on page 237 shows our entry for the parameters mentioned earlier in points one through three.

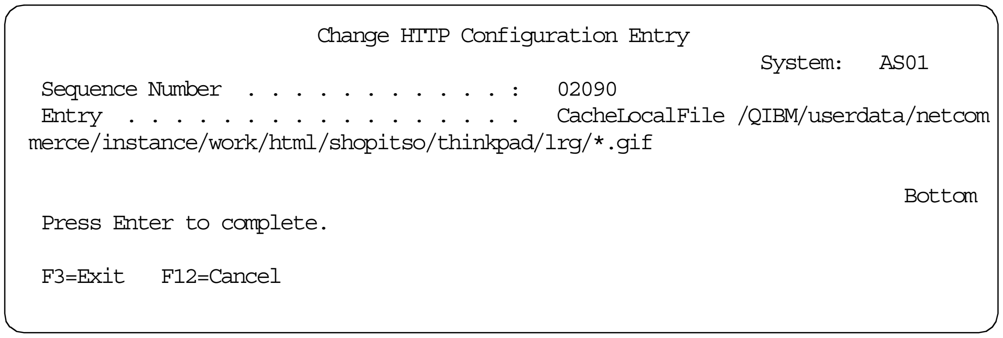


```
                         Work with HTTP Configuration
                                                           System:   AS01
  Configuration name . . . . . . . :   WORK

 Type options, press Enter.
   1=Add   2=Change   3=Copy   4=Remove   5=Display   13=Insert

       Sequence
 Opt    Number    Entry

        02010     CGIConvMode %%EBCDIC%%
        02020     keyfile /QIBM/userdata/icss/cert/server/server.kyr
        02030     ScriptTimeOut 5 minutes
        02040     PersistTimeout 30 seconds
        02050     ServerTerm /QSYS.LIB/QHTTPSVR.LIB/QZHJSVLT.SRVPGM:Adapt   >
        02060     CacheLocalMaxBytes        2 M
        02070     CacheLocalMaxFiles        200
        02080     LiveLocalCache        Off
 2      02090     CacheLocalFile /QIBM/userdata/netcommerce/instance/work   >
        02100     CacheLocalFile /QIBM/userdata/netcommerce/instance/work   >
                                                                     More...
 F3=Exit   F5=Refresh   F6=Print List   F12=Cancel   F17=Top   F18=Bottom
 F19=Edit Sequence
```

Figure 198. Local Cache File Parameter

Figure 199 shows one entry for our .gif files that are stored in the `/qibm/userdata/netcommerce/instance/work/html/shopitso/thinkpad/lrg/` directory.

You can change this entry be entering selection `2` in the Opt field.



```
                        Change HTTP Configuration Entry
                                                           System:   AS01
  Sequence Number  . . . . . . . . . . . :   02090
  Entry  . . . . . . . . . . . . . . . . .   CacheLocalFile /QIBM/userdata/netcom
merce/instance/work/html/shopitso/thinkpad/lrg/*.gif


                                                                        Bottom
  Press Enter to complete.

  F3=Exit   F12=Cancel
```

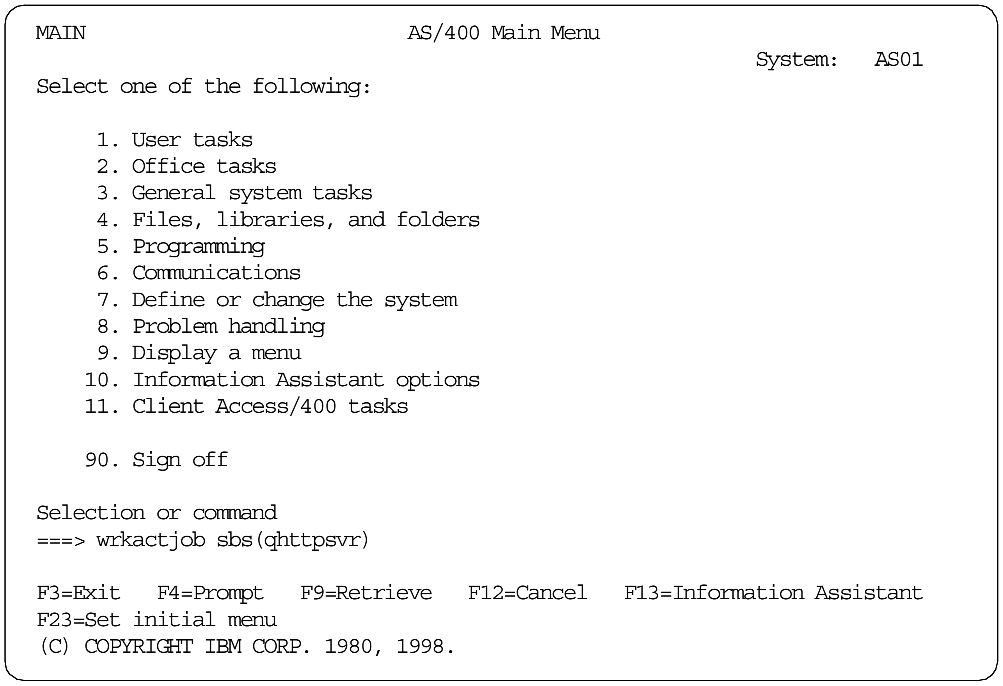
Figure 199. Local Cache File Entry

13.14.1 HTTP Server Trace Output File

To prove if the pages you want to cache in the HTTP server are really cached, you can start the HTTP server with the -vv value. This makes it possible to trace the HTTP server. Use the AS/400 command `STRTCPSVR SERVER (*HTTP) HTTPSVR(<instance_name>)`. Press **PF4** to prompt. Type `-vv` in the Instance startup values field.

To view the HTTP server service trace output file, use the AS/400 commands noted in the following steps:

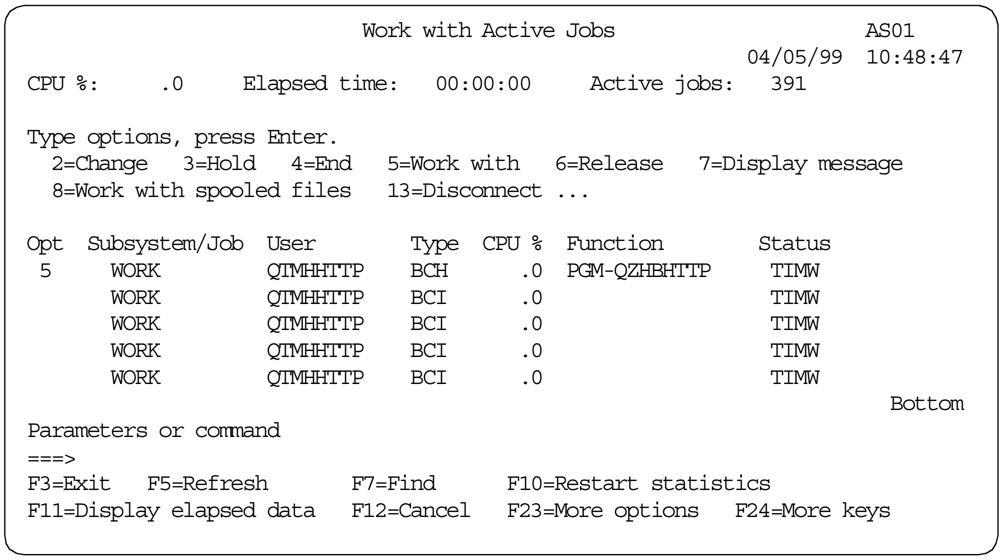
1. Use the `WRKACTJOB` command for the `QHTTPSVR` subsystem to find the job for your HTTP instance.



```
MAIN                              AS/400 Main Menu
                                                             System:   AS01
Select one of the following:

     1. User tasks
     2. Office tasks
     3. General system tasks
     4. Files, libraries, and folders
     5. Programming
     6. Communications
     7. Define or change the system
     8. Problem handling
     9. Display a menu
    10. Information Assistant options
    11. Client Access/400 tasks

    90. Sign off

Selection or command
===> wrkactjob sbs(qhttpsvr)

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel   F13=Information Assistant
F23=Set initial menu
(C) COPYRIGHT IBM CORP. 1980, 1998.
```

Figure 200. Find HTTP Server Job

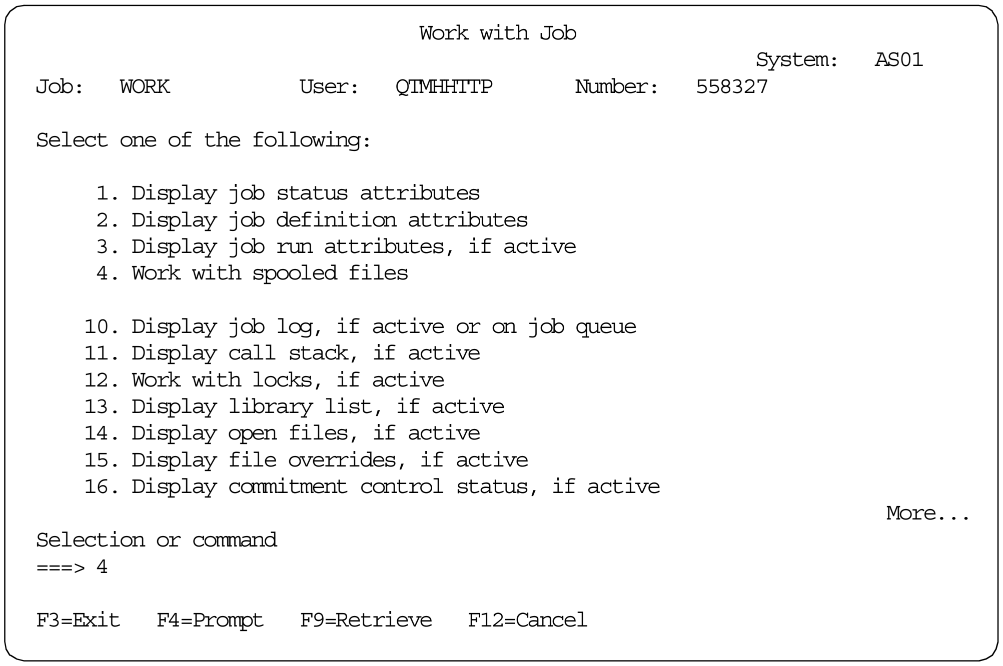
2. Type `5` in the Opt field for the first job that you can find for your HTTP instance. Press **Enter**.



```
                           Work with Active Jobs                    AS01
                                                            04/05/99  10:48:47
CPU %:     .0     Elapsed time:   00:00:00     Active jobs:   391

Type options, press Enter.
  2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
  8=Work with spooled files   13=Disconnect ...

Opt  Subsystem/Job  User        Type  CPU %  Function        Status
 5     WORK         QTMHHTTP    BCH      .0  PGM-QZHBHTTP     TIMW
       WORK         QTMHHTTP    BCI      .0                   TIMW
       WORK         QTMHHTTP    BCI      .0                   TIMW
       WORK         QTMHHTTP    BCI      .0                   TIMW
       WORK         QTMHHTTP    BCI      .0                   TIMW
                                                                        Bottom
Parameters or command
===>
F3=Exit   F5=Refresh       F7=Find      F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F23=More options   F24=More keys
```

Figure 201. Work with HTTP Server Job

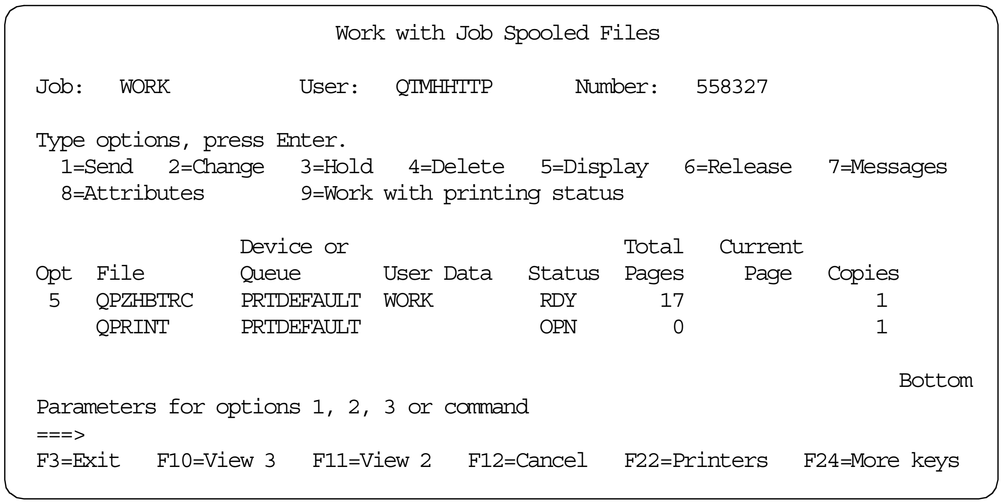
3. Type `4` to work with spooled files for this job. Press **Enter**.



```
                                 Work with Job
                                                            System:   AS01
Job:   WORK          User:   QTMHHTTP      Number:   558327

Select one of the following:

     1. Display job status attributes
     2. Display job definition attributes
     3. Display job run attributes, if active
     4. Work with spooled files

    10. Display job log, if active or on job queue
    11. Display call stack, if active
    12. Work with locks, if active
    13. Display library list, if active
    14. Display open files, if active
    15. Display file overrides, if active
    16. Display commitment control status, if active
                                                                       More...
Selection or command
===> 4

F3=Exit   F4=Prompt   F9=Retrieve   F12=Cancel
```

Figure 202. Start Work with Spooled Files

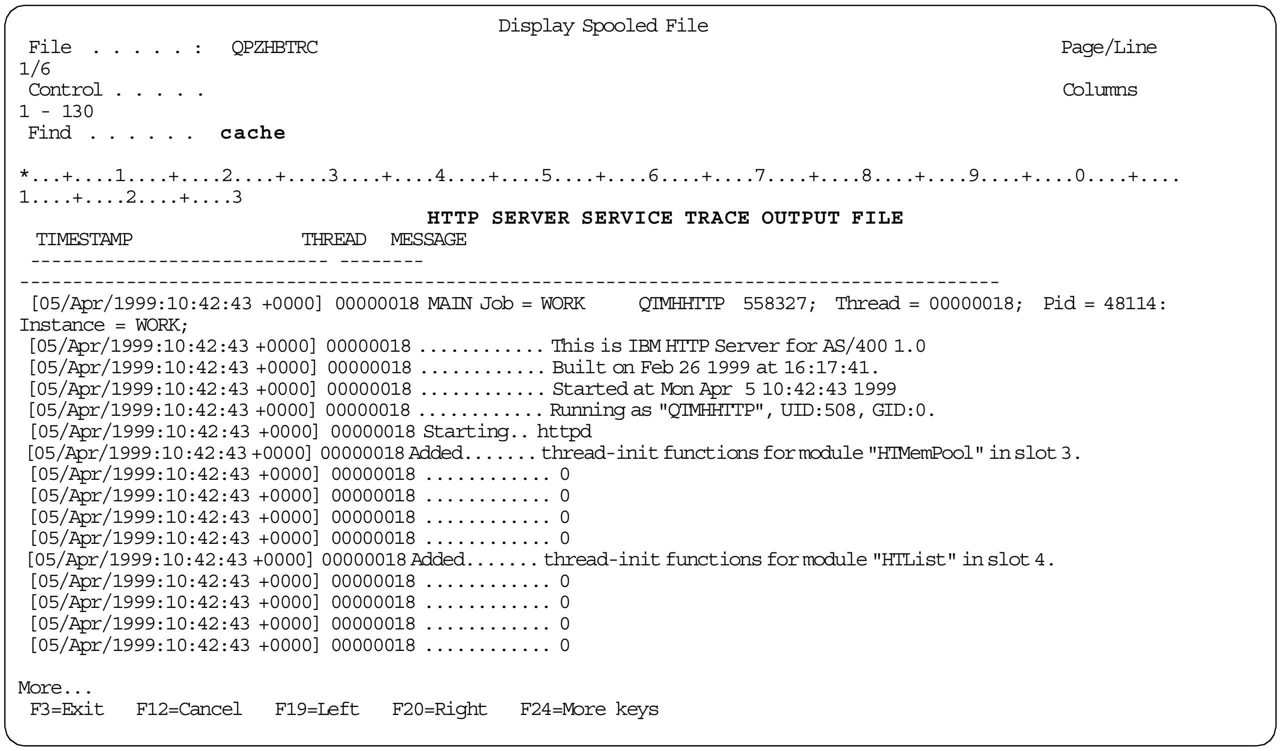
4. Type `5` to display the spooled file. Press **Enter**.



```
                          Work with Job Spooled Files

Job:   WORK          User:   QTMHHTTP      Number:   558327

Type options, press Enter.
  1=Send   2=Change   3=Hold   4=Delete   5=Display   6=Release   7=Messages
  8=Attributes       9=Work with printing status

             Device or                     Total   Current
Opt  File        Queue       User Data   Status  Pages     Page   Copies
 5   QPZHBTRC    PRTDEFAULT  WORK         RDY       17                 1
     QPRINT      PRTDEFAULT               OPN        0                 1

                                                                        Bottom
Parameters for options 1, 2, 3 or command
===>
F3=Exit   F10=View 3   F11=View 2   F12=Cancel   F22=Printers   F24=More keys
```

Figure 203. Start Display Spooled Files

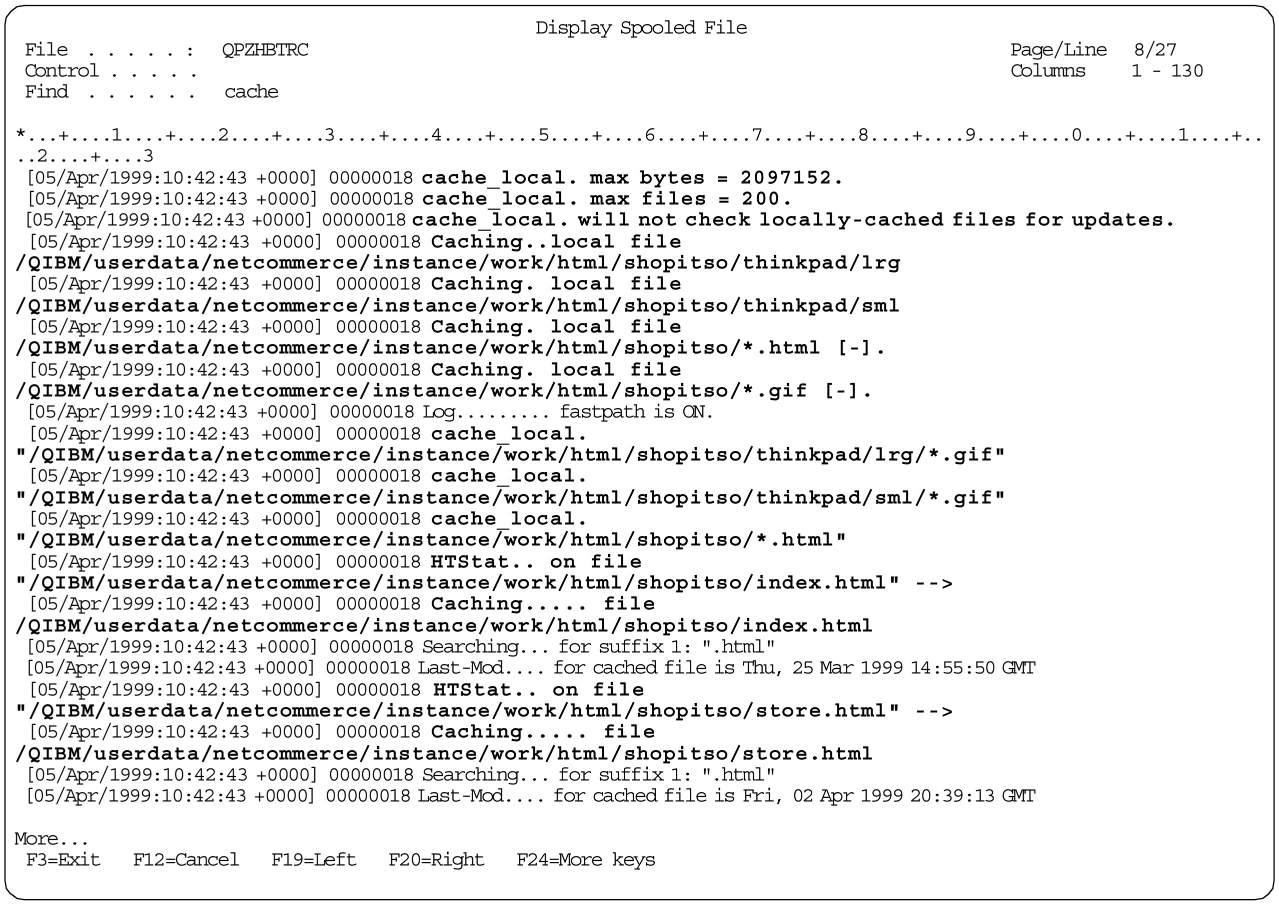
5. Here you see the first site of the HTTP server trace protocol. Type `cache` in the Find field. Use the **PF16** function key to find the protocol notes for the Local cache functions.



```
                                           Display Spooled File
 File  . . . . . :   QPZHBTRC                                                                      Page/Line
1/6
 Control . . . . .                                                                                 Columns
1 - 130
 Find  . . . . . .  cache

*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....+....
1....+....2....+....3
                                       HTTP SERVER SERVICE TRACE OUTPUT FILE
  TIMESTAMP                  THREAD  MESSAGE
 ---------------------------- --------
---------------------------------------------------------------------------------------------
 [05/Apr/1999:10:42:43 +0000] 00000018 MAIN Job = WORK     QTMHHTTP  558327;  Thread = 00000018;  Pid = 48114:
Instance = WORK;
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... This is IBM HTTP Server for AS/400 1.0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... Built on Feb 26 1999 at 16:17:41.
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... Started at Mon Apr  5 10:42:43 1999
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... Running as "QTMHHTTP", UID:508, GID:0.
 [05/Apr/1999:10:42:43 +0000] 00000018 Starting.. httpd
 [05/Apr/1999:10:42:43 +0000] 00000018 Added....... thread-init functions for module "HTMemPool" in slot 3.
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 Added....... thread-init functions for module "HTList" in slot 4.
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0
 [05/Apr/1999:10:42:43 +0000] 00000018 ........... 0

More...
 F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

Figure 204. First site of HTTP Server Trace Protocol

Figure 205 on page 241 shows you the protocol entries for the Local cache function.



```
                                            Display Spooled File
 File  . . . . . :   QPZHBTRC                                                                      Page/Line   8/27
 Control . . . . .                                                                                 Columns     1 - 130
 Find  . . . . . .   cache

*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....+....1....+..
..2....+....3
 [05/Apr/1999:10:42:43 +0000] 00000018 cache_local. max bytes = 2097152.
 [05/Apr/1999:10:42:43 +0000] 00000018 cache_local. max files = 200.
 [05/Apr/1999:10:42:43 +0000] 00000018 cache_local. will not check locally-cached files for updates.
 [05/Apr/1999:10:42:43 +0000] 00000018 Caching..local file
/QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/lrg
 [05/Apr/1999:10:42:43 +0000] 00000018 Caching. local file
/QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/sml
 [05/Apr/1999:10:42:43 +0000] 00000018 Caching. local file
/QIBM/userdata/netcommerce/instance/work/html/shopitso/*.html [-].
 [05/Apr/1999:10:42:43 +0000] 00000018 Caching. local file
/QIBM/userdata/netcommerce/instance/work/html/shopitso/*.gif [-].
 [05/Apr/1999:10:42:43 +0000] 00000018 Log......... fastpath is ON.
 [05/Apr/1999:10:42:43 +0000] 00000018 cache_local.
"/QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/lrg/*.gif"
 [05/Apr/1999:10:42:43 +0000] 00000018 cache_local.
"/QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/sml/*.gif"
 [05/Apr/1999:10:42:43 +0000] 00000018 cache_local.
"/QIBM/userdata/netcommerce/instance/work/html/shopitso/*.html"
 [05/Apr/1999:10:42:43 +0000] 00000018 HTStat.. on file
"/QIBM/userdata/netcommerce/instance/work/html/shopitso/index.html" -->
 [05/Apr/1999:10:42:43 +0000] 00000018 Caching..... file
/QIBM/userdata/netcommerce/instance/work/html/shopitso/index.html
 [05/Apr/1999:10:42:43 +0000] 00000018 Searching... for suffix 1: ".html"
 [05/Apr/1999:10:42:43 +0000] 00000018 Last-Mod.... for cached file is Thu, 25 Mar 1999 14:55:50 GMT
 [05/Apr/1999:10:42:43 +0000] 00000018 HTStat.. on file
"/QIBM/userdata/netcommerce/instance/work/html/shopitso/store.html" -->
 [05/Apr/1999:10:42:43 +0000] 00000018 Caching..... file
/QIBM/userdata/netcommerce/instance/work/html/shopitso/store.html
 [05/Apr/1999:10:42:43 +0000] 00000018 Searching... for suffix 1: ".html"
 [05/Apr/1999:10:42:43 +0000] 00000018 Last-Mod.... for cached file is Fri, 02 Apr 1999 20:39:13 GMT

More...
 F3=Exit   F12=Cancel   F19=Left   F20=Right   F24=More keys
```

Figure 205. Protocol Entries for Local Cache Function

13.15 Modifying Net.Data Macros

As mentioned before, the store that was built with the Store Creator does not meet all of our business requirements. We want to have more functionality. For example, the customer should have the possibility to enter a quantity for the product they want to buy. To implement this functionality, we have to change some Net.Data macros.

To change the Net.Data macros, you need skills in using Net.Data and HTML. You should also know which Net.Commerce commands you need to build the behavior of the shopping process. See 3.8.3, “Mapping the Navigation Flow to Net.Commerce Commands” on page 54.

For information about how to integrate the Net.Commerce commands in a Net.Data macro, see 3.8.2, “Using Net.Commerce Commands” on page 52.

For more information about Net.Data, go to the following sites on the Web:

- For cross-platform information: `http://www.software.ibm.com/data/net.data/`
- For AS/400 information:
  `http://www.as400.ibm.com/tstudio/netdata/news/newfun6.htm`
- For Net.Data literature:
  `http://www.software.ibm.com/data/net.data/library.html`

Important

Net.Commerce can cache Web pages to improve performance. If basic or advance cache support is enabled, copies of the pages are cached in two directories `protect` and `unprotect` under the cache directory `/QIBM/UserData/NetCommerce/instance/<instance_name>/ nc_cache`.

The Synchronization daemon of Net.Commerce recognizes changes that are made in the Net.Commerce DB2/400 database from which the pages are built and automatically deletes the cached files affected.

When you change macros for category of product templates, you must delete each file for the categories or products manually from the cache directory. Otherwise, the cached copy continues to be served and your changes do not take affect.

For more details about the Net.Commerce cache function, refer to *Net.Commerce for AS/400 Installing and Getting Started Guide* (ncinst.pdf), GC09-2864, and *Net.Commerce for AS/400 Net.Commerce Utilities,* (nc_util.pdf) available in softcopy with the Net.Commerce product.

Table 14 describes the changes made to the Net.Data macros that were created from the Store Creator. These changes are necessary to fit our design for our sample ShopITSO store.

Table 14. Net.Data Macros Changes Made for ShopITSO

Net.Data Macro	Reason for Change
cat1.d2w	SQL function GET_SHIPPING_REF_NUM() deleted. Not used here because it is in the prod1.d2w.
prod1.d2w	Possibility to enter quantity.
err_adrbk_uo.d2w	Possibility to work with SET payment function.
order.d2w	Possibility to work with SET payment function SQL function GET_1800NUMBER() deleted SQL DETERMINE_SHIPPING_LIST() changed $(CURRENCY) replaced with $(V_oycpcur)
oderlstc.d2w	New HTML design. New shipped status and shipping date from back-end-system (update from back-end-system in ORDERS table.
orderok.d2w	Add quantity information and total position price in order list New HTML design.
search.d2w	We made a new static HTML page for this window, which is cached through the Local cache function of the AS/400 Web server.
searchrslt.d2w	New SQL statement. We want this search function only for the description field in the product table.
shipto.d2w	Possibility to change quantity.

We created a new Net.Data macro for the Top Category named cat0.d2w. See 13.15.1, “Net.Data Macro to Show the Category Tree” on page 243.

We created the new err_stdata.d2w Net.Data macro to inform the customer that they entered a wrong value in the quantity field. See 13.16, “Exception Handling Conditions by Example” on page 258.

Note

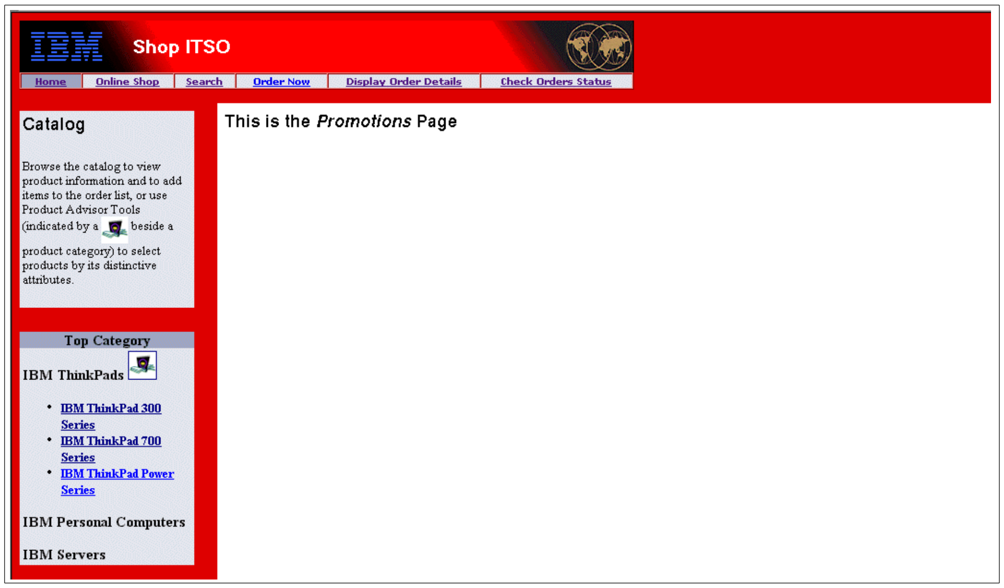
You can use any editor to create or change Net.Data macros. We recommend that you use an editor that shows line numbers for the source code. This is useful when you test your macros because any errors in the macro that occurred are shown with a line number. We used the Net Objects Script Builder software to change the Net.Data macros.

13.15.1 Net.Data Macro to Show the Category Tree

Following our design requirements, we have to build a new macro that shows the three-level deep category structure. See Figure 206.

This Catalog Tree is integrated in the left frame of the first Net.Commerce site. It comes up when the customer starts the Online Shop from the Home Page. See Figure 189 on page 231. From this tree, they can navigate through our product catalog.

The tree shows the second-level categories, in our case, the IBM ThinkPads, IBM Personal Computers, and the IBM Servers categories. The third-level categories are shown directly under the second-level categories. The first-level category, the Top Category, is not shown on this page.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

Top Category

IBM ThinkPads

- IBM ThinkPad 300 Series
- IBM ThinkPad 700 Series
- IBM ThinkPad Power Series

IBM Personal Computers

IBM Servers

This is the *Promotions* Page

Figure 206. First Site in Our ShopITSO with Catalog Tree

To get the data for this hierarchy, we used the following SQL functions in the Net.Data macro:

```
%define {
 SHOWSQL="YES"
 rootTable = %table
 rowIndexRoot = "1"
 parentH1=""
 catChild=""
%}

%{== all categories below top category ==%}
%function(dtw_SQL) GET_CATEGORY_ROOT1(OUT table) {
   SELECT CGRFNBR, CGNAME
   from category, cgryrel
   where  CATEGORY.CGRFNBR = CGRYREL.CRCCGNBR and CATEGORY.CGMENBR = $(MerchantRefNum)
          and CGRYREL.CRPCGNBR = $(HomeCategory) and cgpub=1
   %REPORT {
   %ROW {
   %}
%}
%MESSAGE {
          default: { %} :continue  %}
%}
%{== all categories below root1 category ==%}
%function(dtw_SQL) GET_CATEGORY_CHILD() {
   SELECT CGRFNBR, CGNAME
   from category, cgryrel
   where  CATEGORY.CGRFNBR = CGRYREL.CRCCGNBR and CATEGORY.CGMENBR = $(MerchantRefNum)
          and CGRYREL.CRPCGNBR = $(parentH1) and cgpub=1
   %REPORT {
   %ROW {
    @DTW_ASSIGN(catChild, V_cgrfnbr)
    <TD> </TD>
    <TD> <A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgmenbr=$(MerchantRefNum)&cgrfnbr=$(V_cgrfnbr)"
TARGET="left">$(V_cgname)</A>
    </TD> </TR> <BR>
   %}
%}
%MESSAGE {
          default: { %} :continue  %}
%}

%{== macro for loop ==%}
%macro_function GET_ROOT_INDEX (IN table){
 %while (rowIndexRoot <=  numRows) {

   <TR> <TD> </TD>
   @DTW_ASSIGN(parentH1, @DTW_TB_RGETV(rootTable, rowIndexRoot, "1"))
   <td> <TD colspan=2> @DTW_TB_RGETV(rootTable, rowIndexRoot, "2")</TD> <BR>
   @GET_CATEGORY_CHILD()
   @DTW_ADD(rowIndexRoot, "1", rowIndexRoot)
  %}
%}

%HTML(REPORT) {
<HTML> <HEAD> <TITLE>Document Title</TITLE> </HEAD>
<BODY>
@GET_CATEGORY_ROOT1(rootTable)
@DTW_TB_ROWS(rootTable, numRows)
@GET_ROOT_INDEX(rootTable)
</BODY>
</HTML>
%}
```

We made a copy of the cat1.d2w Net.Data macro and named it cat0.d2w. In this macro, we integrated the above mentioned new SQL functions. Also in this new macro, we integrated the activation for the product advisor. See 14.5, “Publishing Product Advisor Pages” on page 316.

The next step was to assign this cat0.d2w macro to the Top Category. To learn how to do this, see 13.8, “Assigning Templates” on page 206. You can find the final cat0.d2w and cat1.d2w Net.Data macros in A.9, “Net.Data Sample Macros” on page 480.

13.15.2 Finding or Assigning a Net.Data Macro for a Specific Display

To find out which Net.Data macro that is created by the Store Manager is called from a specific task, use the Site Manager function Task Management form. Also when you want to assign your own new macro to a Net.Commerce task, you can follow this description. It works for all Net.Commerce display commands in a similar manner, with the exception of the CategoryDisplay or ProductDisplay pages.

The last mentioned two commands work with the category or product Net.Data template macros. For more information about this template, see 13.8, “Assigning Templates” on page 206.

For example, the following process explains how to find the Net.Data macro name for the `OrderDisplay` Net.Commerce command for a particular stop:

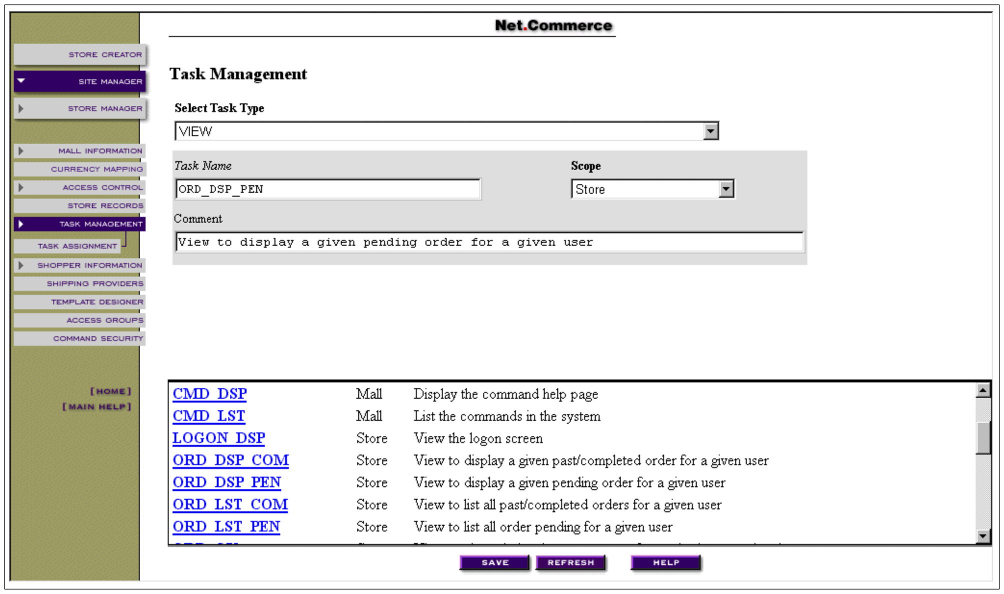
1. Start the Net.Commerce Administration Task with the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

2. Login with your Site Manager user and password or use the `ncadmin` user and choose **Site Manager**.
3. Select **Task Management** on the left side of the window.
4. Select **VIEW** as the *Task Type* in the selection list.
5. In the second half of the left window, search for task **ORD_DSP_PEN** and mark it. You see a window similar to the one shown in Figure 207.

Tip

To get a larger second half of the screen, set the cursor of the black line and move this line above.



Task Management

Select Task Type: VIEW

Task Name: ORD_DSP_PEN

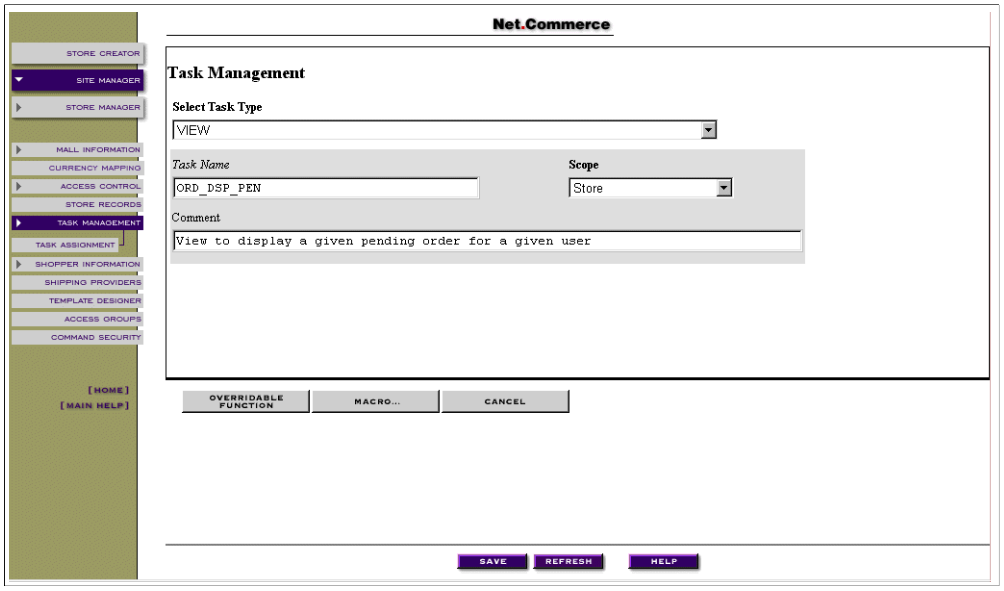
Scope: Store

Comment: View to display a given pending order for a given user

CMD_DSP	Mall	Display the command help page
CMD_LST	Mall	List the commands in the system
LOGON_DSP	Store	View the logon screen
ORD_DSP_COM	Store	View to display a given past/completed order for a given user
ORD_DSP_PEN	Store	View to display a given pending order for a given user
ORD_LST_COM	Store	View to list all past/completed orders for a given user
ORD_LST_PEN	Store	View to list all order pending for a given user

Figure 207. Task Management — View Task

6. Choose **Task Assignment** on the left side of the window. The window appears as shown in Figure 208 on page 246.



Task Management

Select Task Type: VIEW

Task Name: ORD_DSP_PEN

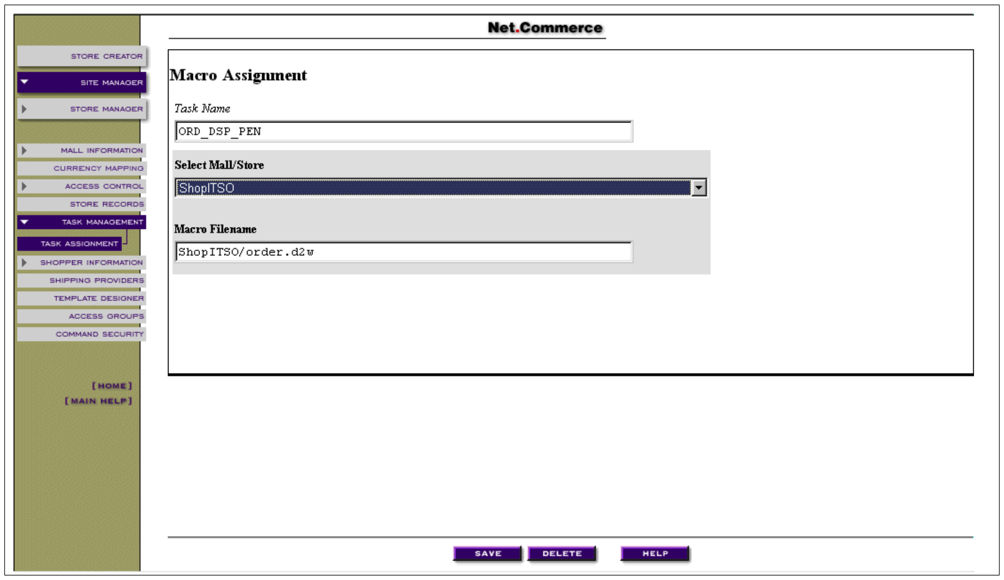
Scope: Store

Comment: View to display a given pending order for a given user

Figure 208. Task Assignment — View Task

7. Click the **MACRO** button. You receive the Macro Assignment window.
8. Select the store name you want to look for in the Select Mall/Store selection list (in our case, ShopITSO).

   You can see the path and macro name in the field Macro Filename as shown in Figure 209.



Macro Assignment

Task Name: ORD_DSP_PEN

Select Mall/Store: ShopITSO

Macro Filename: ShopITSO/order.d2w

Figure 209. Macro Assignment for ORD_DSP_PEN Task

Note

If there is no specific macro for the store, you get the message: `The store currently uses the mall setting for the above task`.

Select **Mall** in the Select Mall/Store selection list to see the macro name for the mall. You can also assign a new Net.Data macro for a specific store in this window.

13.15.3 Original Net.Data Macro for the Product Display

For an example of a Net.Data macro that the Store Creator delivers with the Stop Shop model, consider the macro for the product display shown in the following example. The name is prod1.d2w.

To make it more readable, we highlighted all function calls. All Net.Commerce commands that are in this macro are in italic letters.

You can find all our final Net.Data macros that are used in our ShopITSO in A.9, “Net.Data Sample Macros” on page 480.

```
%include "ShopITSO/ShopITSO.inc"

%{======================================================================

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-D245798-NC3

(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

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

%define {
SHOWSQL="NO"
SHIPPING_REF="0"
ATTRIBUTES = "FALSE"
ITEMS = "FALSE"
ITEM_ATTR_NAME = ""
ADDRESS_REF = ""
%}

%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_SHIPPING_REF_NUM() {
   select mmrfnbr, mmsmnbr, spchrge
   from mshipmode, shipping
   where mmmenbr=$(MerchantRefNum) and spmenbr=mmmenbr and spmmnbr=mmrfnbr
   order by spchrge ASC

   %REPORT{
     %ROW{

%IF (ROW_NUM == "1" && V_mmsmnbr != "0" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_mmrfnbr)
%ELIF (ROW_NUM == "2" && V_mmsmnbr != "0" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_mmrfnbr)
%ELIF (ROW_NUM == "3" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_mmrfnbr)
%ENDIF

     %}
   %}
   %MESSAGE{
     default: {SHIPPING MODE ERROR %}: continue
   %}
%}


%function(dtw_odbc) DISPLAY_BACKUP(){
    select distinct cgname
    from category
    where cgrfnbr=$(CGRY_NUM) and cgmenbr=$(MerchantRefNum)

%REPORT{

<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>

    %ROW{
<FONT COLOR="$(TitleTxtCol)" SIZE=2>
<B><A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(CGRY_NUM)&cgmenbr=$(MerchantRefNum
)">RETURN TO $(V_cgname)</A></B>
</FONT>
    %}

</TABLE>
  %}
  %MESSAGE{100:{%} :continue %}
%}


%function(dtw_odbc) CHECK_PRODUCT_ATTR() {
   selectpdname
   from proddstatr
   where pdprnbr=$(prrfnbr) and pdmenbr=$(MerchantRefNum)

   %REPORT{
     %ROW{
     @DTW_assign(ATTRIBUTES, "TRUE")
     %}
   %}
   %MESSAGE{
     default: { %}: continue
   %}
%}

%function(dtw_odbc) CHECK_ITEMS() {
   selectprrfnbr
   from product
   where prprfnbr=$(prrfnbr) and prmenbr=$(MerchantRefNum)

   %REPORT{
     %ROW{
     @DTW_assign(ITEMS, "TRUE")
     %}
   %}
   %MESSAGE{
     default: { %}: continue
   %}
%}



%function(dtw_odbc) DISPLAY_PRODUCT_BANNER(){
    select prrfnbr, prnbr, prsdesc
    from product
    where prrfnbr=$(prrfnbr) and prmenbr=$(prmenbr)

  %REPORT{

    %ROW{

%IF (ATTRIBUTES == "FALSE" && ITEMS == "FALSE")
<TR>
<TD ALIGN="center" VALIGN="top" BGCOLOR=$(BodyColor2)>
<FONT COLOR=$(TitleTxtCol)><B>$(V_PRSDESC)</B></FONT>
</TD>
</TR>
<TR>
<TD ALIGN="right" VALIGN="top">
<A
HREF="/cgi-bin/ncommerce3/OrderItemUpdate?merchant_rn=$(MerchantRefNum)&product_rn=$(V
_prrfnbr)&quantity=1&shipmode_rn=$(SHIPPING_REF)&url=%2Fcgi-bin%2Fncommerce3%2FOrderIt
emDisplay%3Fmerchant_rn%3D$(MerchantRefNum)" TARGET="list">
<img src="$(AddButton)" border=0></A>
</TD>
</TR>

%ELSE
<TR>
<TD ALIGN="center" VALIGN="top" BGCOLOR=$(BodyColor2) COLSPAN=2>
<FONT COLOR=$(TitleTxtCol) SIZE=3><B>$(V_PRSDESC)</B></FONT>
</TD>
  </TR>
<TR>
<TD ALIGN="center" VALIGN="top" COLSPAN=2 HEIGHT=20>
</TD>
  </TR>
%ENDIF

    %}

  %}
  %MESSAGE{100:{ Product Banner Error%} :continue %}
%}

%function(dtw_odbc) DISPLAY_PRODUCT_PRICE(){

    SELECT prnbr, ppprc, ppcur
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr) and ppprnbr=$(prrfnbr)

  %REPORT{

    %ROW{

<TR><TD><BR></TD></TR>

<TR BGCOLOR="$(BodyColor2)">
<TD ALIGN="center" COLSPAN=2>
<FONT SIZE=2><B>Price : </B>$(V_ppprc) $(V_ppcur)</FONT>
</TD>
</TR>

<TR><TD><BR></TD></TR>

    %}

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

%function(dtw_odbc) DISPLAY_PRODATTR_PRICE(){

    SELECT distinct paname, paval
    FROM PRODUCT, PRODATR, PRODDSTATR
    WHERE pamenbr=$(MerchantRefNum) and prmenbr=$(MerchantRefNum) and paprnbr=prrfnbr and
prprfnbr=$(prrfnbr)
and paname=pdname


  %REPORT{

<FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="list" METHOD="post">
<INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
<INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
<INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
<INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">


<TR><TD><BR></TD></TR>


    %ROW{

%IF (ITEM_ATTR_NAME != V_paname)

</SELECT>
</TD></TR>
@DTW_assign(ITEM_ATTR_NAME, V_paname)
<TR><TD ALIGN="right">
<B>$(V_paname)</B>
</TD>
<TD>
<SELECT NAME="$(V_paname)">
<OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>

%ELSE

<OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>

%ENDIF


    %}

</SELECT>
</TD></TR>
<TR><TD>xxx<INPUT TYPE="text" NAME="quantity" VALUE="1" SIZE="10" MAXLENGTH="32"></TD></TR>

<TR><TD ALIGN="center" COLSPAN=2>yyy<input type=submit value="Add to Order List"></TD></TR>

</FORM>
<TR><TD><BR></TD></TR>

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



%function(dtw_odbc) DISPLAY_PRODITEM_PRICE(){

    SELECT ppprnbr, prnbr, prsdesc, ppprc, ppcur
    FROM product, prodprcs
    WHERE prmenbr=$(MerchantRefNum) and prprfnbr=$(prrfnbr) and ppprnbr=prrfnbr

  %REPORT{

    %ROW{


<TR>
<TD ALIGN="left" BGCOLOR="$(BodyColor2)">
<FONT SIZE=2>
<B>$(V_prsdesc)</B>
<BR>$(V_ppprc) $(V_ppcur)
</FONT>
</TD>
<TD ALIGN="right" VALIGN="top">
<A
HREF="/cgi-bin/ncommerce3/OrderItemUpdate?merchant_rn=$(MerchantRefNum)&product_rn=$(V
_ppprnbr)&quantity=1&shipmode_rn=$(SHIPPING_REF)&url=%2Fcgi-bin%2Fncommerce3%2FOrderIt
emDisplay%3Fmerchant_rn%3D$(MerchantRefNum)" TARGET="list">
<img src="$(AddButton)" border=0></A>
</TD>

</TR>

<TR>
<TD ALIGN="left">
<FONT SIZE=2></FONT>
</TD>
</TR>


    %}

<TR><TD><BR></TD></TR>

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


%function(dtw_odbc) DISPLAY_PRODUCT_IMAGE(){

    SELECT prthmb, prfull
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr)

  %REPORT{

    %ROW{

%IF (V_prfull != "")
<TR>
<TD COLSPAN=2 ALIGN="center">
<IMG SRC="$(V_prfull)">
</TD>
</TR>

%ELIF (V_prthmb != "")
<TR>
<TD COLSPAN=2 ALIGN="center">
<IMG SRC="$(V_prthmb)">
</TD>
</TR>

%ELSE
<TR>
<TD COLSPAN=2 ALIGN="center">
<B><I>Sorry, An image of the product is not available.</I></B>
</TD>
</TR>

%ENDIF

    %}

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


%function(dtw_odbc) DISPLAY_PRODUCT_INFO(){

    SELECT prldesc1, prldesc2, prldesc3
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr)

  %REPORT{

    %ROW{

<TR>
<TD COLSPAN=2>
<FONT SIZE=2>
$(V_prldesc1)
$(V_prldesc2)
```

```
$(V_prldesc3)
</FONT>
</TD>
</TR>

    %}

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



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

%HTML_REPORT{

<HTML>

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

<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">

@GET_ADDRESS_REF_NUM()
@GET_SHIPPING_REF_NUM()

<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTh=100%>
<TR>

<TD ALIGN="left">
<FONT FACE="helvetica" COLOR="$(TitleTxtCol)"><H3>Catalog</H3></FONT>
</TD>
<TD ALIGN="right" VALIGN="top">
%IF (CGRY_NUM != "")
@DISPLAY_BACKUP()
%ENDIF
</TD>


</TR>
<TR>
<TD bleft" COLSPAN=2>
<FONT COLOR="$(TitleTxtCol)" SIZE=2>
Browse the catalog to view product information and to add items to the order list.  </FONT>
<BR><BR>
</TD>
</TR>

</TABLE>

@CHECK_PRODUCT_ATTR()
%IF (ATTRIBUTES == "FALSE")
@CHECK_ITEMS()
%ENDIF

<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTh=100%>

@DISPLAY_PRODUCT_BANNER()

%IF (ATTRIBUTES == "TRUE")

@DISPLAY_PRODUCT_IMAGE()
@DISPLAY_PRODATTR_PRICE()
@DISPLAY_PRODUCT_INFO()
%ELIF (ITEMS == "TRUE")

@DISPLAY_PRODUCT_INFO()
@DISPLAY_PRODUCT_IMAGE()
@DISPLAY_PRODITEM_PRICE()

%ELSE
@DISPLAY_PRODUCT_IMAGE()
@DISPLAY_PRODUCT_PRICE()
```

```
@DISPLAY_PRODUCT_INFO()
%ENDIF

</TABLE>

</BODY>
</HTML>

%}
```

13.15.4 Changes to Net.Data Macro for the Product Display

We made the following changes (in order) in the Net.Data macro for the display product page (name prod1.d2w) to meet our requirements for the sample store:

1. Added the input field for quantity.
2. Improved performance.
3. Changed the frame navigation.
4. Enhanced the ability to get a product price from the back-end system.
5. Added the ability to prove if the quantity in the input field is valid.

Important

When you change macros for the category of the product display pages, you have to delete each file for the concerned categories or products manually from the cache directory. Otherwise, your changes will not take in effect.

Net.Commerce recognizes only changes that are taken in the DB2/400 database. In this case, it deletes the cache file automatically.

The following sequence outlines how we made these changes in the prod1.d2w macro:

1. Added the input field for quantity.

   In all SQL functions that use the Net.Commerce command `OrderItemUpdate`, we have to add an input field to obtain the quantity. This input field has to be in an HTML form tag.

   Because we deleted some SQL functions and built a new structure, we only have to make the changes in the DISPLAY_PRODUCT_IMAGE and DISPLAY_PRODATTR_VALUES SQL functions.

2. Improved performance.

   We deleted some SQL functions. You can find the reason for these changes in Table 15 on page 254. For good performance, we have now only one SQL function, the DISPLAY_PRODUCT_IMAGE, to get the product short description, the name of the product images, and the three fields for the product long description. This was possible because we have no item products in our store.

Table 15. SQL Functions Deleted in the Net.Data Macro

Function Name	Behavior	Reason
GET_SHIPPING_REF_NUM	Delivers a shipping charge	New SQL statement
DISPLAY_BACKUP	Goes back to show the parent category	We work with frames. This link is not necessary.
CHECK_ITEMS	Proves if there are items	We have no items in our product line. We work only with products and products with attributes.
DISPLAY_PRODUCT_BANNER	Delivers product short description from table PRODUCT and calls command OrderItemUpdate, when the product is no item or attribute	We do this in one function, so we integrated this in DISPLAY_PRODUCT_I MAGE. Advantage: Only one SQL query
DISPLAY_PRODUCT_PRICE	Delivers price for products from table PRODPRCS	We get our price from the backend-system, so we do not need this function.
DISPLAY_PRODATTR_PRICE	Delivers product attribute values (not the price, the name of the function is mislead) and calls command OrderItemUpdate, when product is an attribute.	We do this in the function DISPLAY_PRODATTR_ VALUES, which is only called when the product is an attribute.
DISPLAY_PRODITEM_PRICE	Delivers price for products witch are items and calls command OrderItemUpdate, when product is an item.	We get our price from the back-end system, and we have no product items.
DISPLAY_PRODUCT_INFO	Delivers the three long product description fields from table PRODUCT.	We do this in the function DISPLAY_PRODUCT_I MAGE Advantage: Only one SQL query

3. Changed the frame navigation.

   Because we work with frames other than the original Stop Shop Model from the Store Creator, we have to change the HTML TARGET value in the SQL functions DISPLAY_PRODUCT_IMAGE and DISPLAY_PRODATTR_VALUES form TARGET="list" to TARGET=main".

4. Enhanced the ability to get a product price from the back-end system.

   The Net.Commerce command `ProductDisplay` is used to display a product page through the MacroDisplay overridable function.

   This command also calls the `GET_CURRENCY` and `GET_BASE_UNIT_PRC` process tasks. The default MacroDisplay overridable function adds the name-value pair *currency* and *price* variables for use by the Net.Data macro. They are the output parameter of the GET_CURRENCY and GET_BASE_UNIT_PRC tasks.

For the GET_BASE_UNIT_PRC task (and also for the GET_BASE_SPE_PRC task, which is used by the OrderItemDisplay Net.Commerce command, that we use later in our navigation flow to display the content of the current order), we wrote a new overridable function. See 19.4, “Overridable Function by Example” on page 419, to get the product price from the back-end system. We assigned a new overridable function to both GET_BASE_xxx_PRC tasks for our shop through the Side Manager Task Management form.

In the macro for the display product page, we only have to use the currency and price variables to display this price from the back-end system. These variables are used in the HTML Report section.

5. Added the ability to prove if the quantity in the input field is valid.

   All error conditions are done in Net.Commerce through the exception conditions of the Net.Commerce tasks. The Net.Data macro handles only exception conditions that result from the SQL select query.

   To prove if the customer typed a valid value in the quantity input field, we have to use the Net.Commerce exception handling mechanism. You can find how this works in 13.16, “Exception Handling Conditions by Example” on page 258.

13.15.5 Our New Net.Data Macro for Display Product Page

The following code sample shows our new prod1.d2w product template macro. You can find all other used macros in our ShopITSO in A.9, “Net.Data Sample Macros” on page 480.

```
%include "ShopITSO/ShopITSO.inc"
%{=====================================================================
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-D245798-NC3

(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
=====================================================================%}
%define {
  SHOWSQL="YES"
SHIPPING_REF="0"
ATTRIBUTES = "FALSE"
ITEM_ATTR_NAME = ""
ADDRESS_REF = ""
      DESC1=""
       DESC2=""
       DESC3=""
%}

%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_SHIPPING_REF_NUM() {
   select spmmnbr, spchrge
   from    shipping
   where spmenbr=$(MerchantRefNum)
   order by spchrge ASC

   %REPORT{
     %ROW{
      %IF (ROW_NUM == "1" && SHIPPING_REF == "0")
  @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ELIF (ROW_NUM == "2" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ELIF (ROW_NUM == "3" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ENDIF
     %}
   %}
   %MESSAGE{
     default: {SHIPPING MODE ERROR %}: continue
   %}
%}

%function(dtw_odbc) CHECK_PRODUCT_ATTR() {
   selectpdname
   from proddstatr
   where pdprnbr=$(prrfnbr) and pdmenbr=$(MerchantRefNum)
   %REPORT{
     %ROW{
     @DTW_assign(ATTRIBUTES, "TRUE")
     %}
   %}
   %MESSAGE{
     default: { %}: continue
   %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_IMAGE(){
    SELECT prthmb, prfull, prsdesc, prldesc1, prldesc2, prldesc3, prnbr
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr)
  %REPORT{
    %ROW{
       <TR BGCOLOR="$(BodyColor2)"><TD><FONT SIZE=2>
          <B>$(V_PRSDESC)</TD>
     <TD ALIGN="left" >
       <B>SKU: $(V_prnbr)</B></FONT></TD></TR>
     <tr> <TD> </TD> </TR>  <tr> <TD> </TD> </TR>
     %IF (V_prfull != "")
     <TR><TD COLSPAN=2 ALIGN="left"><IMG SRC="$(V_prfull)"></TD></TR>
     %ELIF (V_prthmb != "")
     <TR><TD COLSPAN=2 ALIGN="left"><IMG SRC="$(V_prthmb)"></TD></TR>
        %ELSE
     <TR><TD COLSPAN=2 ALIGN="left"><B><I>Sorry, An image of the product is not
available.</I></B></TD></TR>
        %ENDIF
     @DTW_assign(DESC1, V_prldesc1)
     @DTW_assign(DESC2, V_prldesc2)
     @DTW_assign(DESC3, V_prldesc3)
     %}
%IF (ATTRIBUTES == "FALSE")
   <TR>
   <FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="main" METHOD="post">
       <TD> Please type quantity you want to order:
       <INPUT TYPE=text NAME=quantity VALUE=1 SIZE=5 MAXLENGTH=32> </TD>
    <INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
    <INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
    <INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
    <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
    <TD ALIGN="left" COLSPAN=2><input type=image SRC="$(AddButton)"></TD></TR>
      </FORM>
<TR><TD><BR></TD></TR>
%ENDIF
  %}
  %MESSAGE{100:{ %} :continue %}
%}



%function(dtw_odbc) DISPLAY_PRODATTR_VALUES(){
    SELECT distinct paname, paval
    FROM PRODUCT, PRODATR, PRODDSTATR
    WHERE pamenbr=$(MerchantRefNum) and prmenbr=$(MerchantRefNum) and paprnbr=prrfnbr and
prprfnbr=$(prrfnbr)
and paname=pdname
  %REPORT{
     <TR>
     <FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="main" METHOD="post">
       <TD> Please type quantity you want to order:
       <TD> <INPUT TYPE=text NAME=quantity VALUE=1 SIZE=5 MAXLENGTH=32> </TD>
    <INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
    <INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
    <INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
    <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
       <BR></TD></TR>
     %ROW{
  %IF (ITEM_ATTR_NAME != V_paname)
</SELECT>
@DTW_assign(ITEM_ATTR_NAME, V_paname)
<TR><TD ALIGN="right"><B>$(V_paname)</B> </TD>
 <TD>
 <SELECT NAME="$(V_paname)"><OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>
 %ELSE
 <OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>
 %ENDIF
    %}
</SELECT>
</TD></TR>
<TR><TD><BR></TD></TR>

<TR><TD ALIGN="center" COLSPAN=2><input type=image SRC="$(AddButton)"></TD></TR>
</FORM>
<TR><TD><BR></TD></TR>
  %}
  %MESSAGE{100:{ PROBLEM%} :continue %}
%}

%{=================================================%}
%{ HTML Report Section
%{=================================================%}
%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY>
@GET_ADDRESS_REF_NUM()
@GET_SHIPPING_REF_NUM()
@CHECK_PRODUCT_ATTR()
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0  WIDTH=100%>
@DISPLAY_PRODUCT_IMAGE()
%IF (ATTRIBUTES == "TRUE")
 @DISPLAY_PRODATTR_VALUES()
%ENDIF
<TR BGCOLOR="$(BodyColor2)"><TD ALIGN="center"><FONT SIZE=2><B>Price : $(price)
$(currency)</FONT></B> </FONT>
</TD> </TR><P>
<TR> <TD COLSPAN=2> <FONT SIZE=2> $(DESC1)<BR> $(DESC2)<BR> $(DESC3)<BR>
</TABLE>
</BODY>
</HTML>
%}
```

13.16 Exception Handling Conditions by Example

To prove if the given value in the quantity input field of the Display Product page is valid, we use the Net.Commerce exception conditions handling mechanism. This exception conditions handling to occur in the same manner for all errors that are proved, as described here for this sample:

1. Find out which exception conditions your used Net.Commerce command provides (see the handbook *Commands, Tasks, Overridable Functions, and Database Tables*, which comes with the Net.Commerce product).
2. Look at the description for the corresponding exception task to find out the behavior (in the same above mentioned handbook).
3. Create a new Net.Data macro for the error page that should be shown to the customer or use an existing one. Perhaps you do not have to make changes. Use the error code and the available name-value pairs.

   Consider the choices the customer has in the error page and consider the page that should be shown next (after the error page) and use the corresponding Net.Commerce command to show this page.
4. Assign the Net.Data macro to the corresponding exception task for your shop or for the whole mall.

Now, we go back to our problem to prove if the value of the quantity is valid. The Net.Commerce command `ProductDisplay` has no exception conditions.

`ProductDisplay` only shows an HTML page (a dynamic page in this case), which has an HTML form tag, that gets input values from the browser. Therefore, the prove mechanism (exception handling) can only be in the next interaction with the server. In our case, the ProductDisplay page calls the `OrderItemUpdate` Net.Commerce command together with the URL `OrderItemDisplay`. See 3.8.3, “Mapping the Navigation Flow to Net.Commerce Commands” on page 54.

The `OrderItemUpdate` command has an exception condition that calls the `BAD_ST_DATA` exception task, if the quantity specified is not numeric or a positive value (in our case, this value comes from the product page), which handles the error.

Look at the description of the `BAD_ST_DATA` exception task in the *Commands, Tasks, Overridable Functions and Database* handbook. In this handbook, you will find that the task delivers an ERROR_CODE. The string value is 220, and the name of the invalid field in string variable field. This error task also generates a page that will be displayed through the `TaskDisplay` in this case, instead of the OrderItemDisplay. This page is also a Net.Data page, which is assigned to the `BAD_ST_DATA` task for the mall or shop.

Now determine which Net.Data macro is assigned to the `BAD_ST_DATA` exception task. To learn how to do this, see 13.16.2, “Assigning a Net.Data Macro to an Exception Task” on page 261.

Take that macro and copy it to your store macro directory (see the following example), when it is not already there. Make the necessary changes on it. See 13.16.1, “Changes in the err_stdata.d2w Macro” on page 259.

Your store macro directory is:

`/QIBM/UserData/NetCommerce/instance/instance_name/macro/shop_name/`

For our store, it is:

`/QIBM/UserData/NetCommerce/instance/work/macro/ShopITSO/`

If you only created a store with the Store Creator Stop Shop model, you will not find any macro for the `BAD_ST_DATA` exception. The Store Creator does not create this macro for you. The Stop Shop model works without a user input field for quantity, so it is not necessary to have it normally there. You have to create the macro on your own.

If you created any store in the mall with the Store Creator and the Personal Delivery store model, you will find the path (the store name in this case) and the err_stdata.d2w name for the Net.Data macro for the BAD_ST_DATA exception task in the corresponding store. If you installed the Metropolitan mall or other Net.Commerce demo stores, you can find an association for these stores too.

If you did not find any macro association to the `BAD_ST_DATA` exception task in your mall, you can also search for a sample of the Net.Data err_stdata.d2w macro in the following path:

`/QIBM/ProdData/macro/MRIx/ncsample`

Or, search for it in the `/QIBM/UserData/` or in the `/QIBM/ProdData/` directory. Copy this macro to your store macro directory.

Note

MRI`xxxx` is the language code of the Net.Commerce license program. If you installed the English version, it is MRI2924. Replace `xxxx` with your language code.

You can also use our err_stdata.d2w macro as described in the following section, which discusses the important changes you can make.

The last step is to assign the new Net.Data macro to the error task `BAD_ST_DATA` for your store through the Task Management form of the Site Manager function. For more information, see 13.16.2, “Assigning a Net.Data Macro to an Exception Task” on page 261.

13.16.1 Changes in the err_stdata.d2w Macro

Keep in mind that the error page should show the customer the error reason and should have a possibility to go further with the shopping flow. This page should call the next possible page, so that the customer can go forward in the application. In our solution, the customer gets the current order list when they click the Go Further button (Net.Commerce command OrderItemDisplay).

Be sure that you make the following changes in the error macro:

- Use the right include file with the name of your store. For our store, the name is ShopITSO/ShopITSO.inc.
- Have the same GET_SHOPPER_REF_NUM SQL function.

- To display also a image as we do (you can use any image), type the correct path name. We used the image from the ncsample directory /ncsample/warning.gif and copied it to our ShopITSO html directory:

  Your store macro directory is:

  `/QIBM/UserData/NetCommerce/instance/instance_name/html/shop_name/`

  For our store, it is:

  `/QIBM/UserData/NetCommerce/instance/work/html/ShopITSO/`

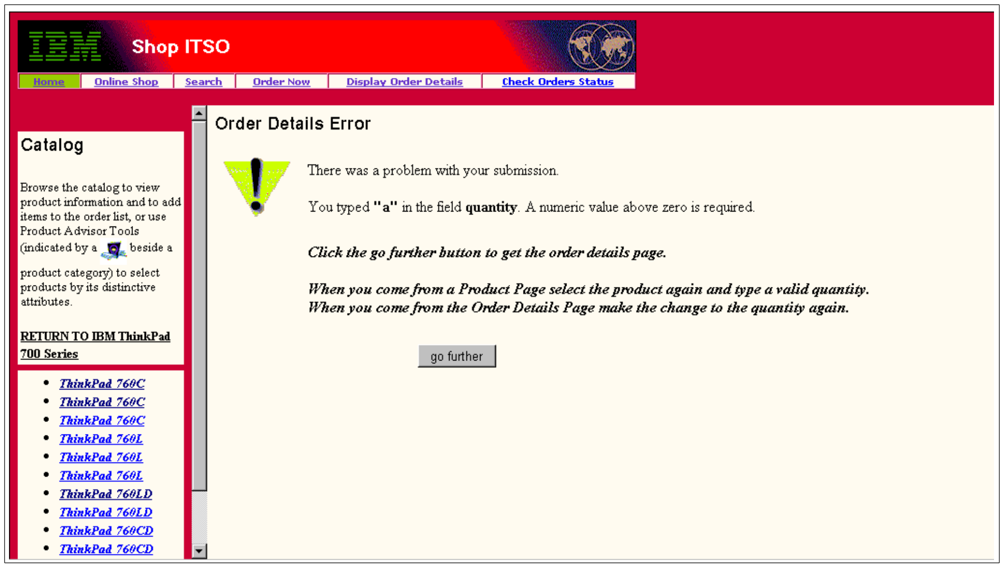
- Use the error code (220 in our case) and the name-value pair (variable field and quantity).

Our err_stdata.d2w macro follows here:

```
%include "ShopITSO/ShopITSO.inc"


%define {
 SHOWSQL="NO"
%}

%{==== Retrieves the Shopper Reference Number ====%}

%b) GET_SHOPPER_REF_NUM() {
   select shrfnbr from shopper where shlogid = '$(SESSION_ID)'
   %REPORT{
     %ROW{
       @DTW_assign(SHOPPER_REF, V_shrfnbr)
     %}
   %}
   %MESSAGE{
     default: { ERROR in GET_SHOPPER_REF_NUM %}
   %}
%}

%HTML_REPORT{
<HTML>

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

<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">

<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR>
      <TD ALIGN="left" VALIGN="center">
  <FONT FACE="helvetica" COLOR=$(TitleTxtCol)><H3>Order Details Error</H3></FONT>
  </TD>
    </TR>
</TABLE>
@GET_SHOPPER_REF_NUM()


<TABLE>
  <TR>
<TD align=center width=85>
  <img src="/ShopITSO/warning.gif" valign=bottom align=left width=72 height=63>
</TD>
<TD>
  There was a problem with your submission.
  <br><P>
  %if ("$(error_code)" == "220")
You typed <B>"$(quantity)"</B> in the field <B>$(field)</B>.  A numeric value above zero is
required.
  %endif
</TD>
  </TR>
  <TR>
<TD align=center width=85>
</TD>
  <TD>
```

```
<BR>
<B><I>Click the go further button to get the order details page.</I><BR><BR>
   <I>When you come from a Product Page select the product again and type a valid
quantity.</I><BR>
   <I>When you come from the Order Details Page make the change to the quantity again.</I></b>
  </TD>
  </TR>
</TABLE>

    <BR>

<TABLE WIDTH=300>
<TR>
  <FORM ACTION="/cgi-bin/ncommerce3/OrderItemDisplay">
<INPUT TYPE="hidden" NAME="merchant_rn" VALUE="$(MerchantRefNum)">
<TD WIDTH=100 ALIGN="right"><INPUT TYPE="submit" VALUE="go further">
</TD>
  </FORM>
</TR>
</TABLE>

</BODY>

</HTML>

%}
```

The page that is generated through this macro appears similar to the example shown in Figure 210. This page is shown when the shopper types in a wrong character in the input field for the quantity and the corresponding submit button (in our case, in the add button from the Product page or the update button in the Display Current Order page).



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad 700 Series

- ThinkPad 760C
- ThinkPad 760C
- ThinkPad 760C
- ThinkPad 760L
- ThinkPad 760L
- ThinkPad 760L
- ThinkPad 760LD
- ThinkPad 760LD
- ThinkPad 760CD
- ThinkPad 760CD

Order Details Error

There was a problem with your submission.

You typed **"a"** in the field **quantity**. A numeric value above zero is required.

Click the go further button to get the order details page.

When you come from a Product Page select the product again and type a valid quantity.
When you come from the Order Details Page make the change to the quantity again.

go further

Figure 210. Bad Quantity Exception Page

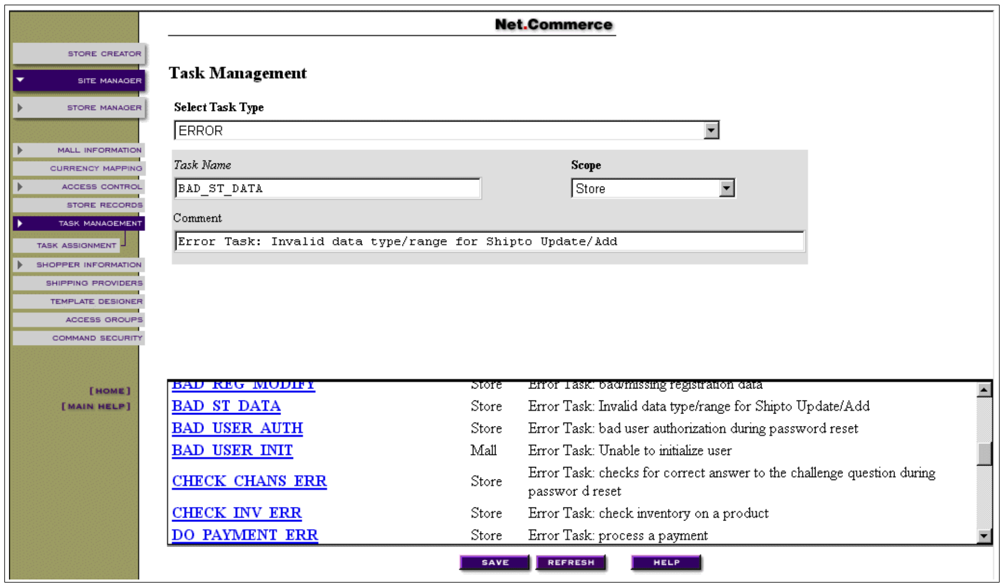
13.16.2 Assigning a Net.Data Macro to an Exception Task

This section explains how to assign a Net.Data macro to an Net.Commerce exception task to display information for the shopper. We assign our err_stdata.d2w macro that we built to inform the customer that they have to enter a valid quantity to the error task BAD_ST_DATA through a Task Management form of the Site Manager function. Follow this process:

1. Start the Net.Commerce Administration Task with the following address:

   http://*fullyqualifiedhost_name*/ncadmin/

2. Login with your Site Manager user and password or use the `ncadmin` user and choose the **Site Manager** task.
3. Select **Task Management** on the left side.
4. Select **ERROR** as the *Task Type* in the selection list.
5. In the second half of the left window, search for the task **BAD_ST_DATA** and mark it. The window, as shown in Figure 211, appears.



Task Management

Select Task Type: ERROR

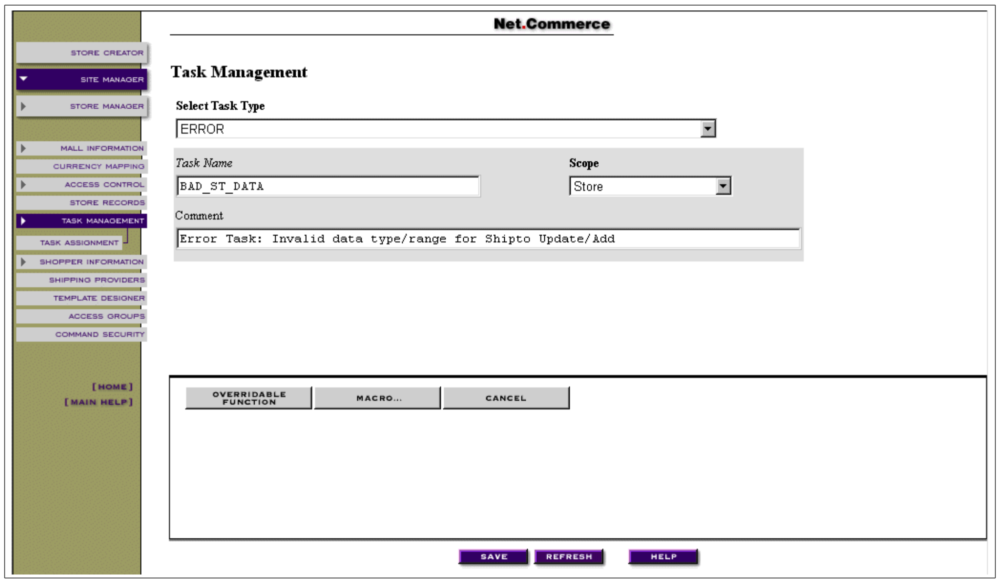
Task Name: BAD_ST_DATA
Scope: Store

Comment: Error Task: Invalid data type/range for Shipto Update/Add

BAD_REG_MODIFY	Store	Error Task: bad/missing registration data
BAD_ST_DATA	Store	Error Task: Invalid data type/range for Shipto Update/Add
BAD_USER_AUTH	Store	Error Task: bad user authorization during password reset
BAD_USER_INIT	Mall	Error Task: Unable to initialize user
CHECK_CHANS_ERR	Store	Error Task: checks for correct answer to the challenge question during passwor d reset
CHECK_INV_ERR	Store	Error Task: check inventory on a product
DO_PAYMENT_ERR	Store	Error Task: process a payment

Figure 211. Task Management — Error Task

6. Choose **Task Assignment** on the left side of the window. The window shown in Figure 212 appears.



Task Management

Select Task Type: ERROR

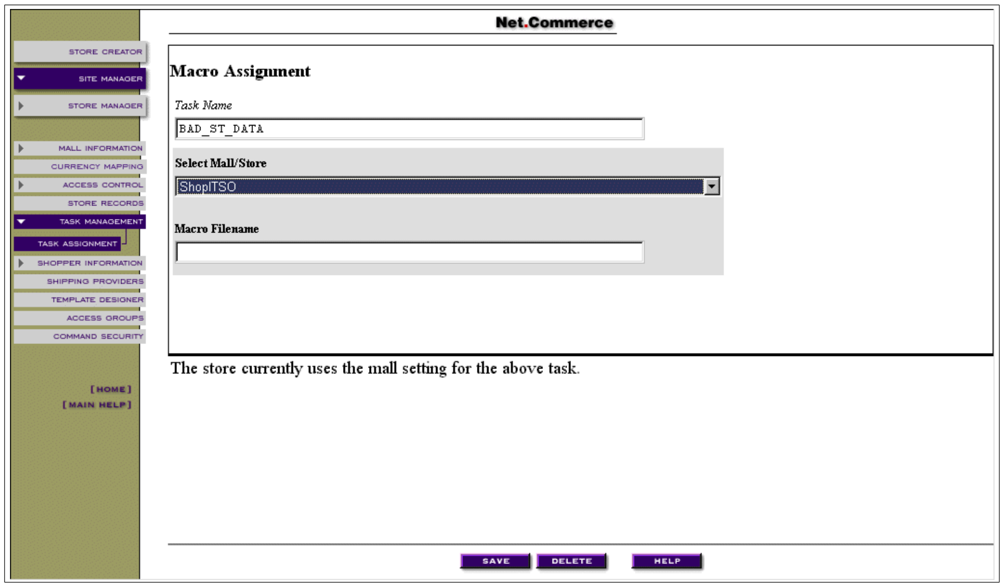
Task Name: BAD_ST_DATA
Scope: Store

Comment: Error Task: Invalid data type/range for Shipto Update/Add

Figure 212. Task Management — Error Task Assignment

7. Click the **MACRO** button. The Macro Assignment window appears.

8. Select the store name you want to look for in the Select Mall/Store selection list (in our case, ShopITSO). If there is no assignment for your store, the text shown in Figure 213 appears.



Macro Assignment

Task Name: BAD_ST_DATA

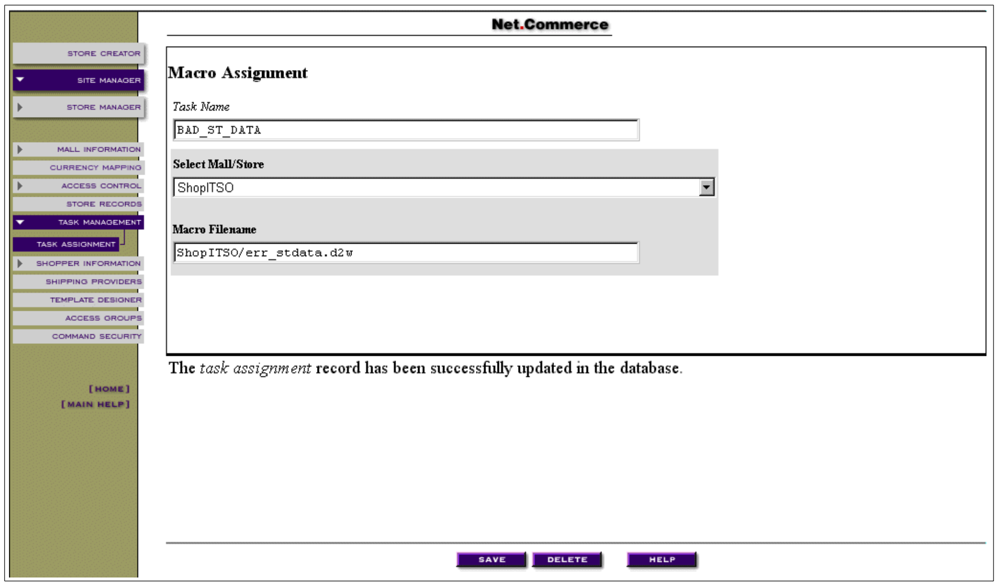
Select Mall/Store: ShopITSO

Macro Filename:

The store currently uses the mall setting for the above task.

Figure 213. Macro Assignment — Task Management Page

9. To assign the macro name for your store, type in the path and macro name for your store in the Macro Filename field. In our case, it is `ShopITSO/err_stdata.d2w`. Click the **Save** button. The window shown in Figure 214 appears with the message indicating that the update was completed successfully.



Macro Assignment

Task Name: BAD_ST_DATA

Select Mall/Store: ShopITSO

Macro Filename: ShopITSO/err_stdata.d2w

The *task assignment* record has been successfully updated in the database.

Figure 214. Macro Assignment for the New Net.Data Macro

13.17 Assigning SSL Protocol to Net.Commerce Commands

A major concern of all users of the World Wide Web is security. This is especially important for commercial transactions containing personal data or credit card details.

Security protocols such as Secure Socket Layer (SSL) protect data sent across the Internet by addressing the following concerns:

- **Confidentiality** — Message content remains private.
- **Integrity** — Messages are not altered while being transmitted.
- **Accountability** — Both parties agree that the exchange happened.
- **Authenticity** — Both parties trust each other's identity.

Net.Commerce uses the SSL protocol to secure transactions:

- RSA public key technology
- The concept of keys to encrypt and decrypt messages

SSL communication is a two-phase process:

1. Handshaking
   - Determine the identity of the other party.
   - Negotiate how the data is going to be sent.
2. Transfer information
   - A key exchange takes place.
   - A browser uses a key to encrypt or decrypt messages.
   - A server receives data and decrypts it, or encrypts data and sends it.

The Command Security function in Net.Commerce Administrator allows you to view and update whether SSL security or authentication (logon) is required to run certain commands. The server automatically re-directs the shopper's browser to a new URL if the requested URL does not begin with https: or if the shopper is not logged in. We did not change any security assignment in our ShopITSO sample.

To view or update the security assignment, follow these steps:

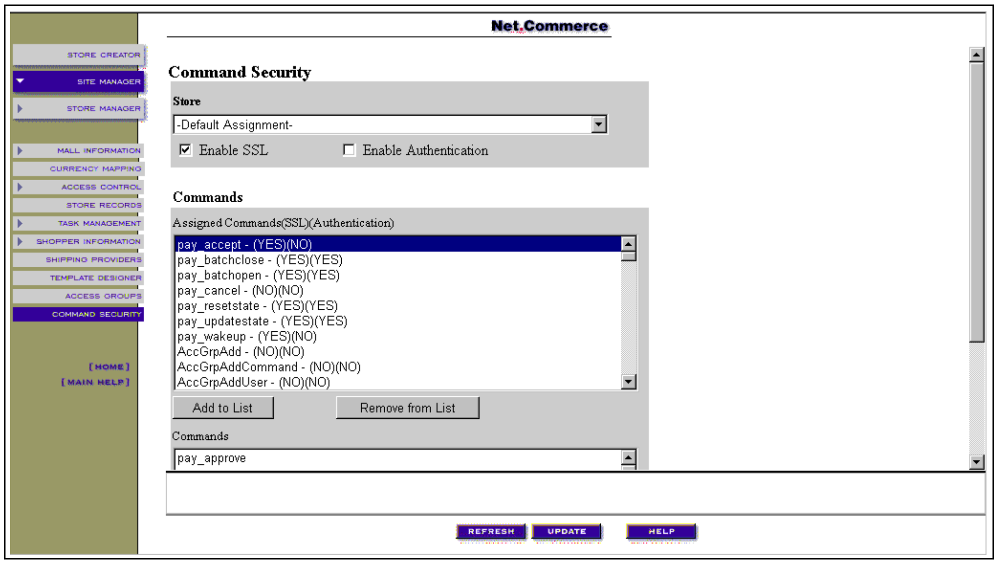
1. Login as `ncadmin` in the Net.Commerce Administration Web page at the following URL:

   http://*fullyqualifiedhost_name*/ncadmin/

2. Select **SITE MANAGER** in the left frame of the window.
3. Select **COMMAND SECURITY** in the left frame of the window.

   A window appears as shown in Figure 215 on page 265.

   In the selection list Store, you see Default Assignment. This means that in the Assigned Commands list (SSL) (Authentication), you see the delivered security assignment to the Net.Commerce commands. This assignment is used for all stores when the store itself has no assignment.



Command Security

Store: -Default Assignment-

[✓] Enable SSL [ ] Enable Authentication

Commands

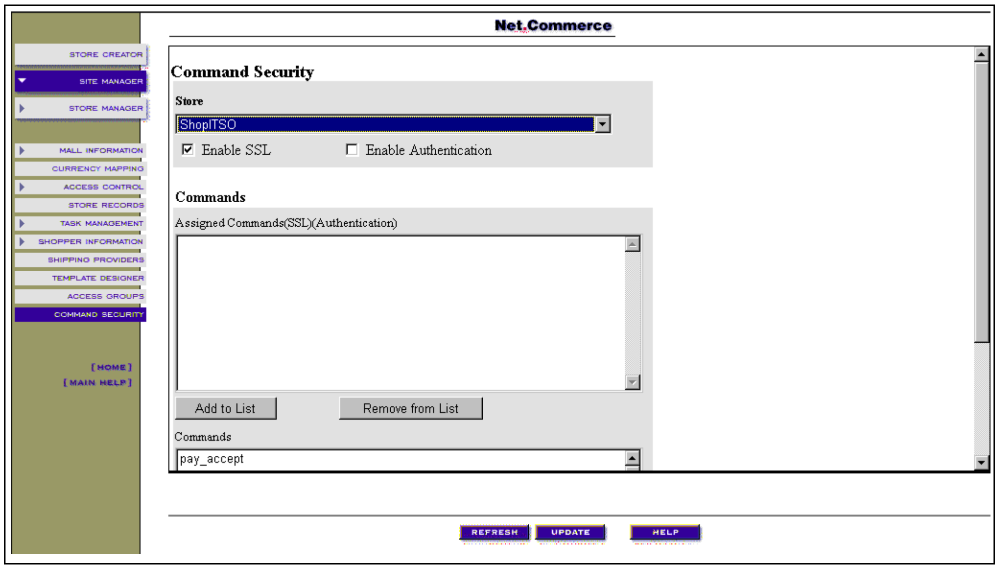
Assigned Commands(SSL)(Authentication)

```
pay_accept - (YES)(NO)
pay_batchclose - (YES)(YES)
pay_batchopen - (YES)(YES)
pay_cancel - (NO)(NO)
pay_resetstate - (YES)(YES)
pay_updatestate - (YES)(YES)
pay_wakeup - (YES)(NO)
AccGrpAdd - (NO)(NO)
AccGrpAddCommand - (NO)(NO)
AccGrpAddUser - (NO)(NO)
```

Commands: pay_approve

Figure 215. Command Security (Part 1 of 7)

4. You can update your store when you choose your store name in the Store selection list.



Command Security

Store: ShopITSO

[✓] Enable SSL [ ] Enable Authentication

Commands

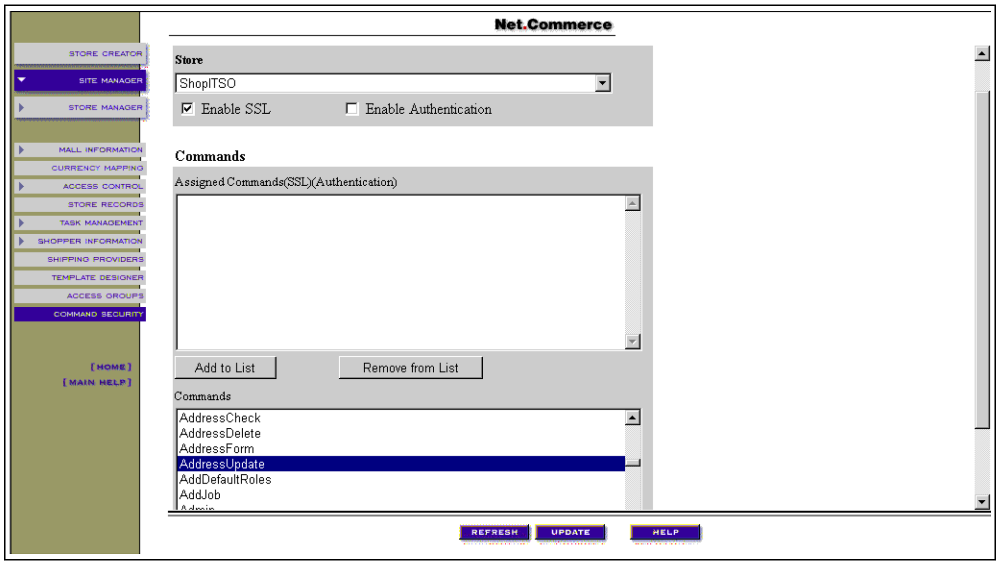
Assigned Commands(SSL)(Authentication)

Commands: pay_accept

Figure 216. Command Security (Part 2 of 7)

The Assigned Commands (SSL) (Authentication) is not filled in. See Figure 216.

5. Choose a Net.Commerce command from the Commands selection list, for example, **Address Update,** as shown in Figure 217 on page 266.



Store

ShopITSO

☑ Enable SSL ☐ Enable Authentication

Commands

Assigned Commands(SSL)(Authentication)

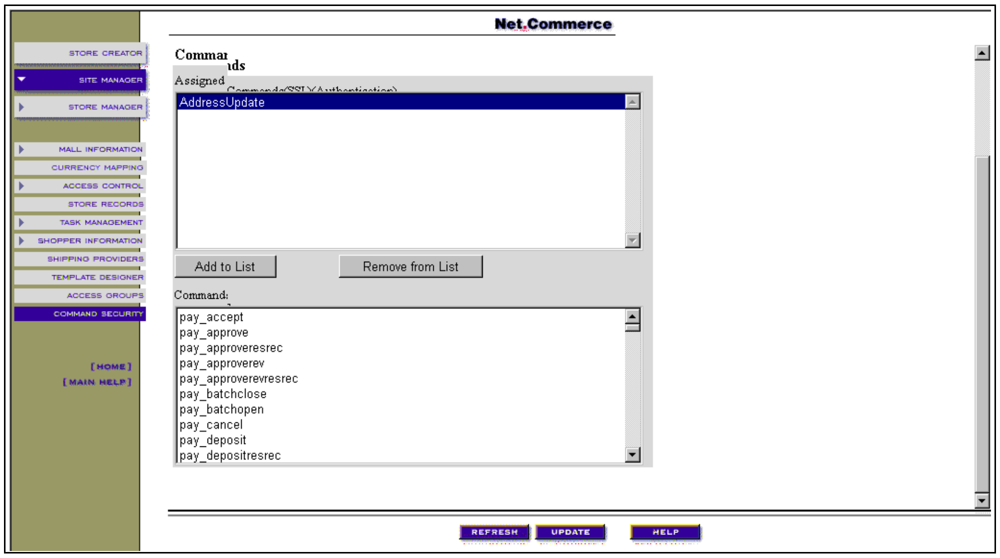
Add to List Remove from List

Commands

AddressCheck
AddressDelete
AddressForm
AddressUpdate
AddDefaultRoles
AddJob
Admin

Figure 217. Command Security (Part 3 of 7)

6. Click the **Add to List** button. A window appears similar to the example in Figure 218.



Commands

Assigned Commands(SSL)(Authentication)

AddressUpdate

Add to List Remove from List

Commands

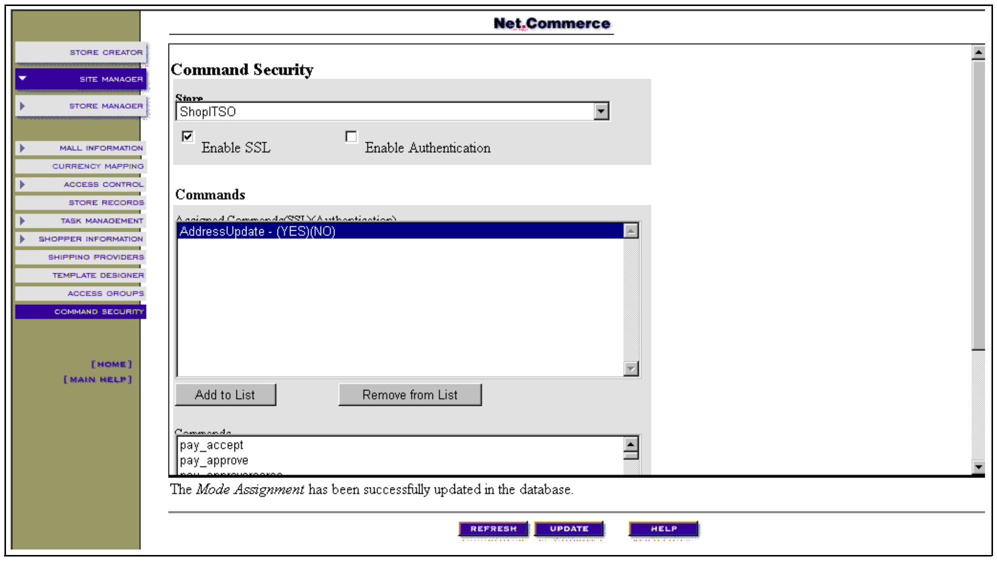
pay_accept
pay_approve
pay_approveresrec
pay_approverev
pay_approverevresrec
pay_batchclose
pay_batchopen
pay_cancel
pay_deposit
pay_depositresrec

Figure 218. Command Security (Part 4 of 7)

7. Click the **UPDATE** button.

   A window similar to the one shown in Figure 219 on page 267 appears. It indicates that the update was completed successfully.

   Notice that the Authentication is set to "No". This is because we have not marked Enable Authentication.



Command Security

Store
ShopITSO

☑ Enable SSL ☐ Enable Authentication

Commands

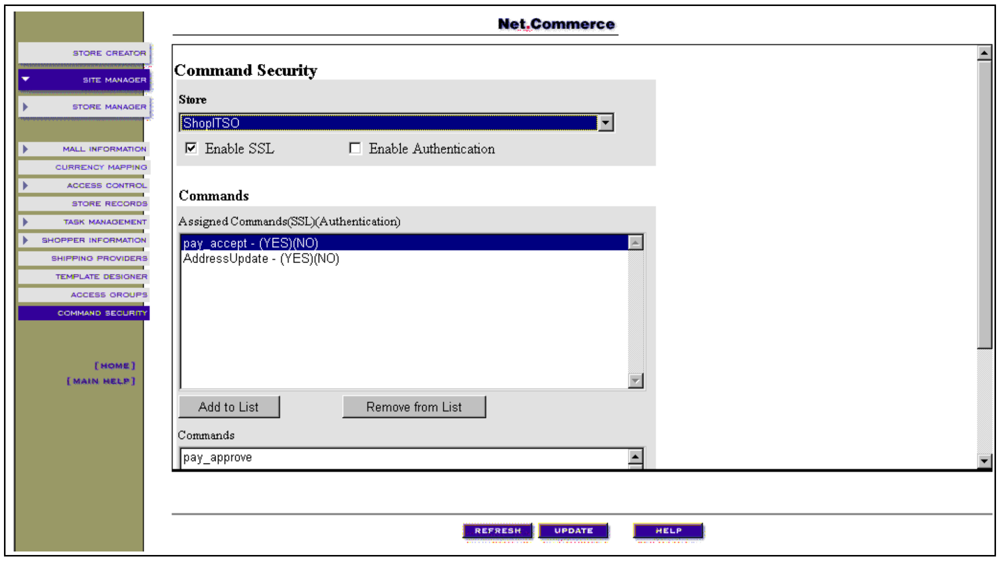
Assigned Commands(SSL)(Authentication)
AddressUpdate - (YES)(NO)

Commands
pay_accept
pay_approve

The *Mode Assignment* has been successfully updated in the database.

Figure 219. Command Security (Part 5 of 7)

We added a second Net.Commerce command (pay_accept) and made the update. You can see the list that results in Figure 220.



Command Security

Store
ShopITSO

☑ Enable SSL ☐ Enable Authentication

Commands

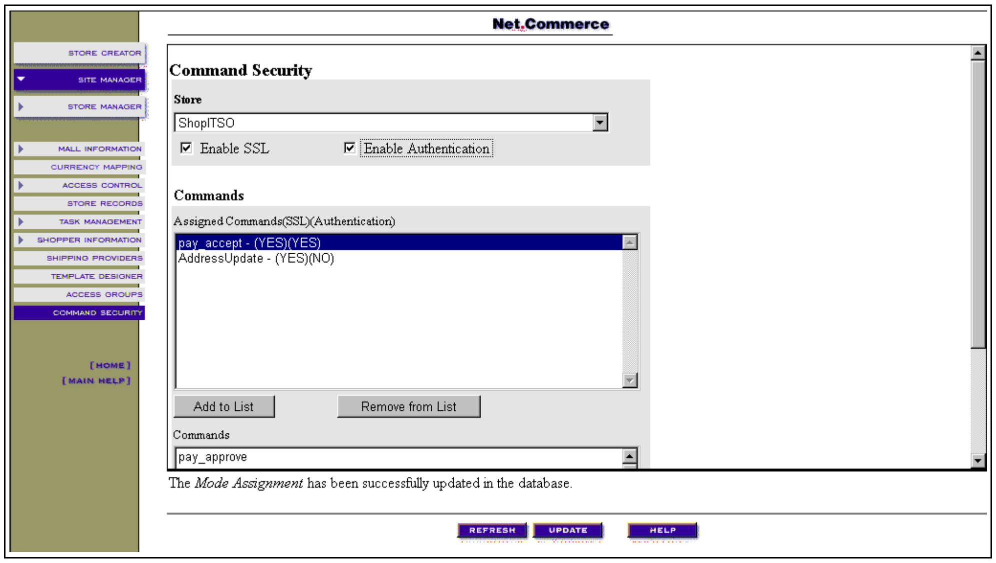
Assigned Commands(SSL)(Authentication)
pay_accept - (YES)(NO)
AddressUpdate - (YES)(NO)

Commands
pay_approve

Figure 220. Command Security (Part 6 of 7)

8. When you also want to enable Authentication (Shopper must logon with a user ID and password), mark the Net.Commerce command you want to change.
9. Check the **Enable Authentication** box and click the **UPDATE** button.

   Now, you have the two Net.Commerce commands with different security assignments. This is shown in Figure 221 on page 268.



Command Security

Store

ShopITSO

☑ Enable SSL ☑ Enable Authentication

Commands

Assigned Commands(SSL)(Authentication)

pay_accept - (YES)(YES)
AddressUpdate - (YES)(NO)

Commands

pay_approve

The *Mode Assignment* has been successfully updated in the database.

Figure 221. Command Security (Part 7 of 7)

After you make your changes, end and start the HTTP server instance. Use these AS/400 commands to do this:

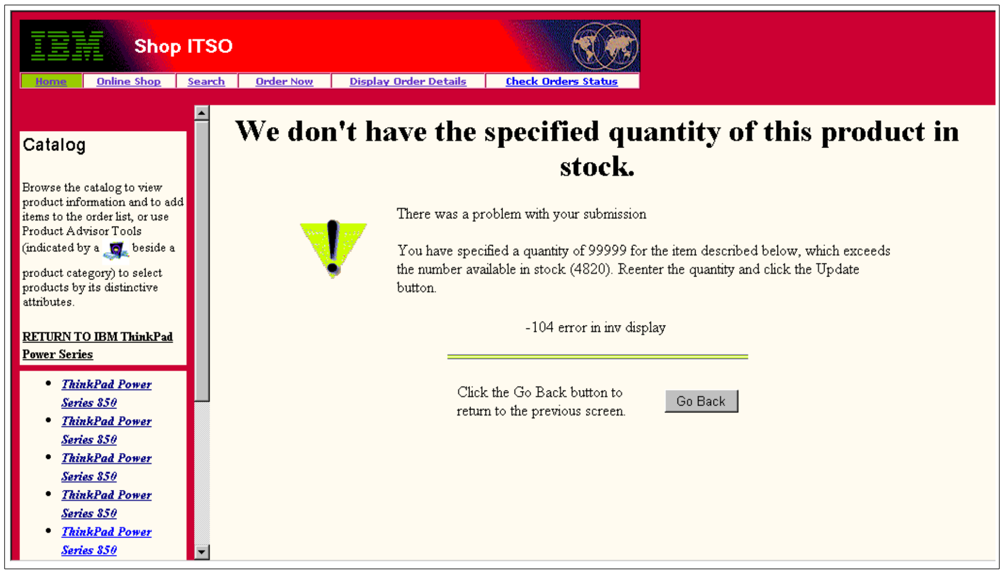
```
ENDTCPSVR SERVER (*HTTP) HTTPSVR(<instance_name>)
STRTCPSVR SERVER (*HTTP) HTTPSVR(<instance_name>)
```

13.18 Disabling Check Inventory

The `CHECK_INV` process task is called from the Net.Commerce commands `OrderDisplay`, `OrderItemProcess`, and `OrderItemUpdate`.

The default implementation of the Net.Commerce `CHECK_INV` process task determines whether there is enough inventory in stock to cover a request for a given quantity of a given product. If not, it sets the `CHECK_INV_ERR` exception task, and handles the exception by writing an HTTP response (shows the window that is built by the assigned macro to the `CHECK_INV_ERR` exception task, the err_check.inv.d2w macro). This macro is delivered through the Store Creator.

Figure 222 on page 269 displays the message that is shown to the customer when the quantity orders are not available.



We don't have the specified quantity of this product in stock.

There was a problem with your submission

You have specified a quantity of 99999 for the item described below, which exceeds the number available in stock (4820). Reenter the quantity and click the Update button.

-104 error in inv display

Click the Go Back button to return to the previous screen.

Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

RETURN TO IBM ThinkPad Power Series

- ThinkPad Power Series 850
- ThinkPad Power Series 850
- ThinkPad Power Series 850
- ThinkPad Power Series 850
- ThinkPad Power Series 850

Figure 222. Out of Stock Message

Because we want to accept any ordered quantity in our ShopITSO e-business application and the order fulfillment is done by our back-end system, we have to disable this behavior.

You can do this by assigning the overridable function `DoNothingNoArgs` to the `CHECK_INV` process task. To do this, follow these steps:

1. Login as `ncadmin` in the Net.Commerce Administration Web page at the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

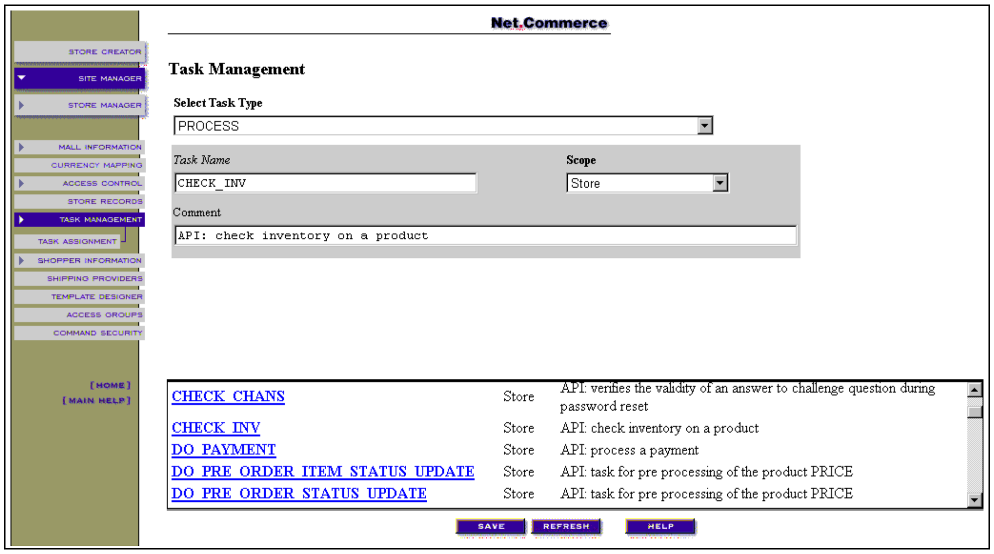
2. Select **SITE MANAGER** in the left frame of the window.
3. Select **TASK MANAGEMENT** in the left frame of the window.
4. Select **PROCESS** in the Select Task Type selection list.

Tip

To get a larger second half of the screen, set the cursor of the black line and move this line above.

5. Search and select **CHECK INV** in the second part of the window.

   The window shown in Figure 223 on page 270 appears.



Task Management

Select Task Type: PROCESS

Task Name: CHECK_INV

Scope: Store

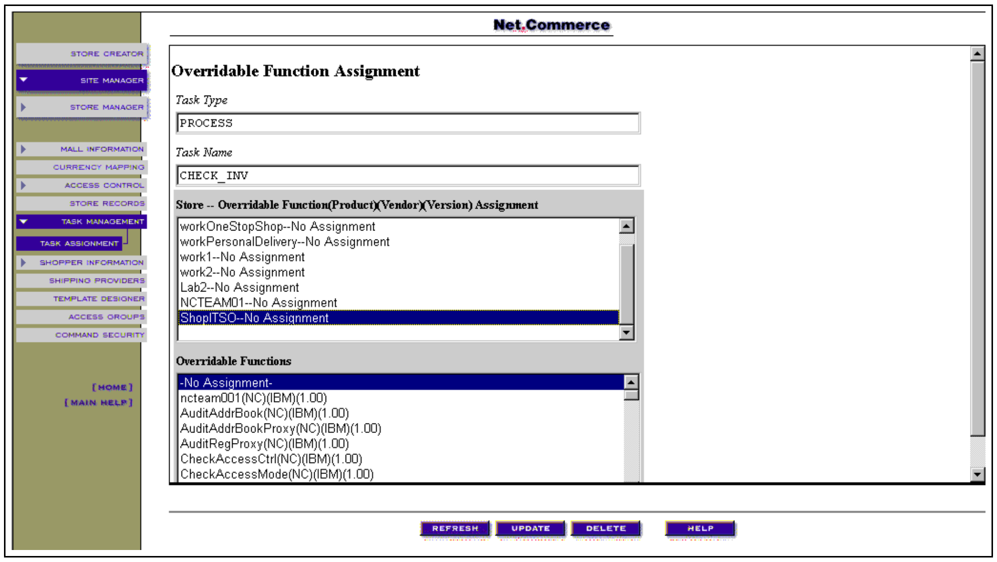
Comment: API: check inventory on a product

CHECK_CHANS	Store	API: verifies the validity of an answer to challenge question during password reset
CHECK_INV	Store	API: check inventory on a product
DO_PAYMENT	Store	API: process a payment
DO_PRE_ORDER_ITEM_STATUS_UPDATE	Store	API: task for pre processing of the product PRICE
DO_PRE_ORDER_STATUS_UPDATE	Store	API: task for pre processing of the product PRICE

Figure 223. Task Management — CHECK_INV (Part 1 of 4)

6. Select **TASK ASSIGNMENT** in the left frame of the window.

   The window shown in Figure 224 appears. You can see that for ShopITSO, there is "No Assignment". This means that the value that is assigned for the mall is also used for our shop.



Overridable Function Assignment

Task Type: PROCESS

Task Name: CHECK_INV

Store -- Overridable Function(Product)(Vendor)(Version) Assignment

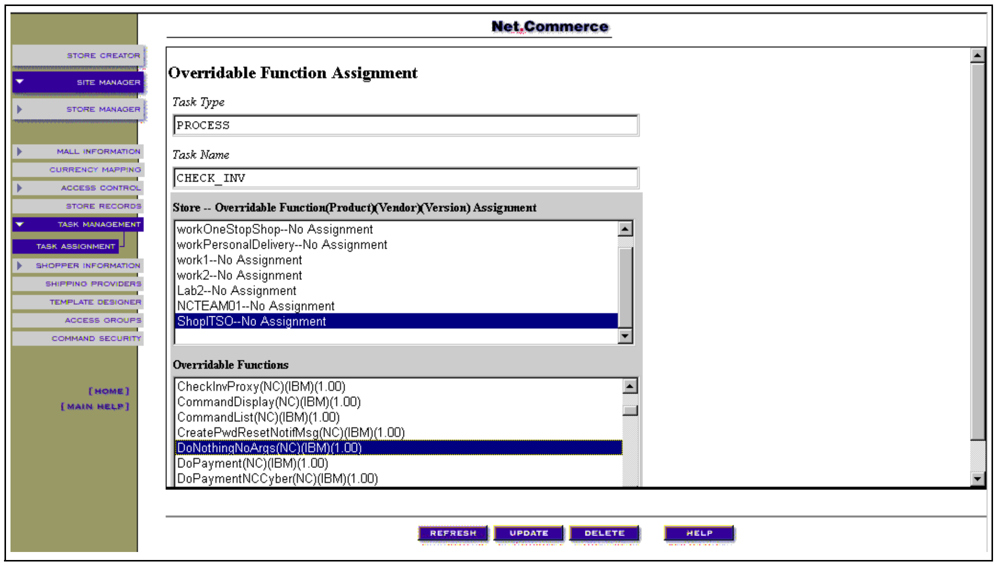
```
workOneStopShop--No Assignment
workPersonalDelivery--No Assignment
work1--No Assignment
work2--No Assignment
Lab2--No Assignment
NCTEAM01--No Assignment
ShopITSO--No Assignment
```

Overridable Functions

```
-No Assignment-
ncteam001(NC)(IBM)(1.00)
AuditAddrBook(NC)(IBM)(1.00)
AuditAddrBookProxy(NC)(IBM)(1.00)
AuditRegProxy(NC)(IBM)(1.00)
CheckAccessCtrl(NC)(IBM)(1.00)
CheckAccessMode(NC)(IBM)(1.00)
```

Figure 224. Task Management — CHECK_INV (Part 2 of 4)

7. Select **DoNothingNoArgs** in the Overridable Functions selection list. See Figure 225 on page 271.



Overridable Function Assignment

Task Type

PROCESS

Task Name

CHECK_INV

Store -- Overridable Function(Product)(Vendor)(Version) Assignment

```
workOneStopShop--No Assignment
workPersonalDelivery--No Assignment
work1--No Assignment
work2--No Assignment
Lab2--No Assignment
NCTEAM01--No Assignment
ShopITSO--No Assignment
```

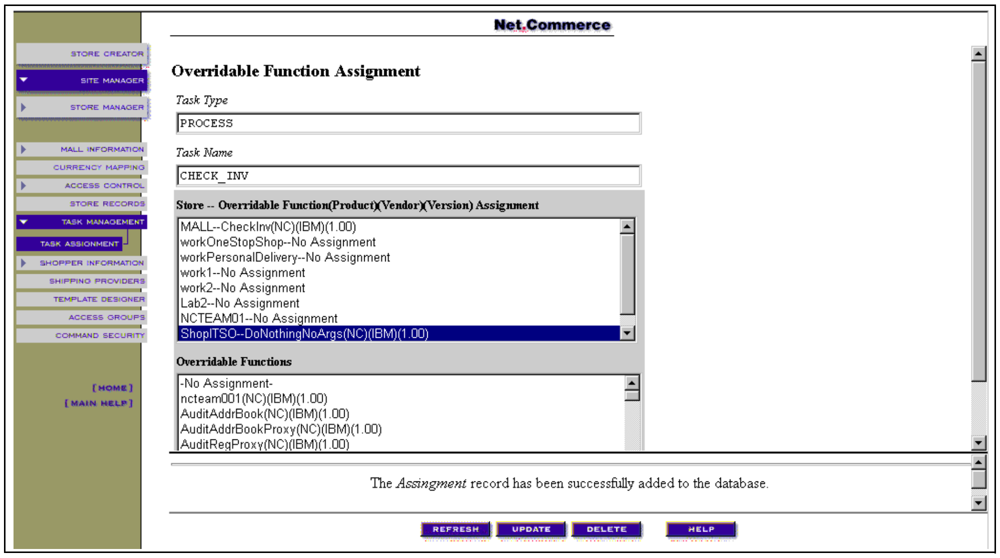
Overridable Functions

```
CheckInvProxy(NC)(IBM)(1.00)
CommandDisplay(NC)(IBM)(1.00)
CommandList(NC)(IBM)(1.00)
CreatePwdResetNotifMsg(NC)(IBM)(1.00)
DoNothingNoArgs(NC)(IBM)(1.00)
DoPayment(NC)(IBM)(1.00)
DoPaymentNCCyber(NC)(IBM)(1.00)
```

Figure 225. Task Management — CHECK_INV (Part 3 of 4)

8. Click the **UPDATE** button.

   If the update is done successfully, the window shown in Figure 226 appears.



Overridable Function Assignment

Task Type

PROCESS

Task Name

CHECK_INV

Store -- Overridable Function(Product)(Vendor)(Version) Assignment

```
MALL--CheckInv(NC)(IBM)(1.00)
workOneStopShop--No Assignment
workPersonalDelivery--No Assignment
work1--No Assignment
work2--No Assignment
Lab2--No Assignment
NCTEAM01--No Assignment
ShopITSO--DoNothingNoArgs(NC)(IBM)(1.00)
```

Overridable Functions

```
-No Assignment-
ncteam001(NC)(IBM)(1.00)
AuditAddrBook(NC)(IBM)(1.00)
AuditAddrBookProxy(NC)(IBM)(1.00)
AuditRegProxy(NC)(IBM)(1.00)
```

The *Assignment* record has been successfully added to the database.

Figure 226. Task Management — CHECK_INV (Part 4 of 4)

9. Stop and restart the Net.Commerce instance through the AS/400 Task function or use the following AS/400 commands from a 5250 session:

```
ENDNETCSVR NetCommerce_instance-name
STRNETCSVR NetCommerce_instance-name
```

   For our store that works with the instance work, it is:

```
ENDNETCSVR WORK
STENETCSVR WORK
```

13.19 Customizing System Error Pages

The term *System Errors* converts a number of types of problem from incorrect user input to complete system failure. These errors result in either:

- *Messages appended to the server logs*. These usually relate to problems starting the server daemons. They can be caused by configuration errors, socket problems or database connectivity problems. Changing the logging level of the server also causes messages to be appended to the server logs.
- *HTML error pages sent to the shopper*. These can either be caused by the director responding to invalid syntax or can relate to more severe problems with the server or database.

The same system problem (for example, the server cannot connect to the database) may result in a log entry and for each user who tried to use the Net.Commerce system a system error page. Figure 227 shows the system error flow.

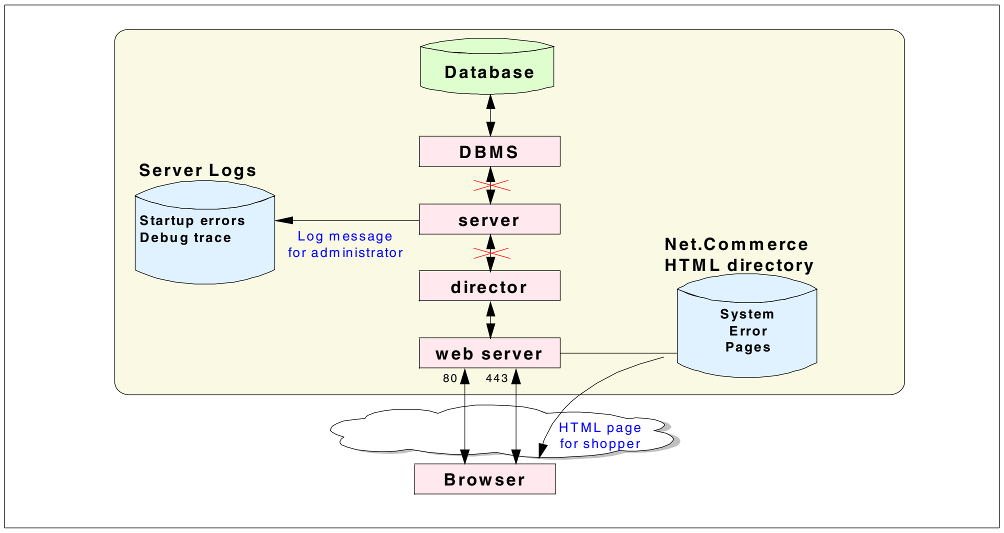


Figure 227. System Error Flow

The system error pages are static HTML files stored in the `QIBM/ProdData/NetCommerce/html/MRIxxxx/ncerror` directory. The directory is mapped in the HTTP Web server configuration file to URL /ncerror.

Each page corresponds to a particular system error. Some of these can indicate a user error or typing mistake in a URL of the store. Other pages may indicate a server or database failure. Table 16 shows the reason, filename, and message number of the system error pages.

Table 16. Table System Error Pages

Message Number	File Name	Reason
CMN0302E	noserver.html	Server not responding Director cannot connect to the server
CMN0950E	cmdinc.html	Command Structure Failure Command has been mistyped or used without the correct parameters
CMN0953E	datapop.html	Data Population Failure a value sent to certain commands doesn't match a row in the DB
CMN0957E	sql.html	SQL Failure The DB manager may be down or the Net.Commerce DB may be corrupt.
CMN0958E	config.html	Configuration Failure API DLL/function cannot be found or if the director cannot communicate with the server
CMN0959E	oom.html	Memory Failure
CMN0960E	env.html	Environment Failure when there is a missing parameter in mserver.ini
CMN0961E	cmdexe.html	Command Execution Failure when a command cannot execute or complete
CMN0962E	auth.html	Authorization Failure when a user tries to access an Administrator page that they do not have authority for.

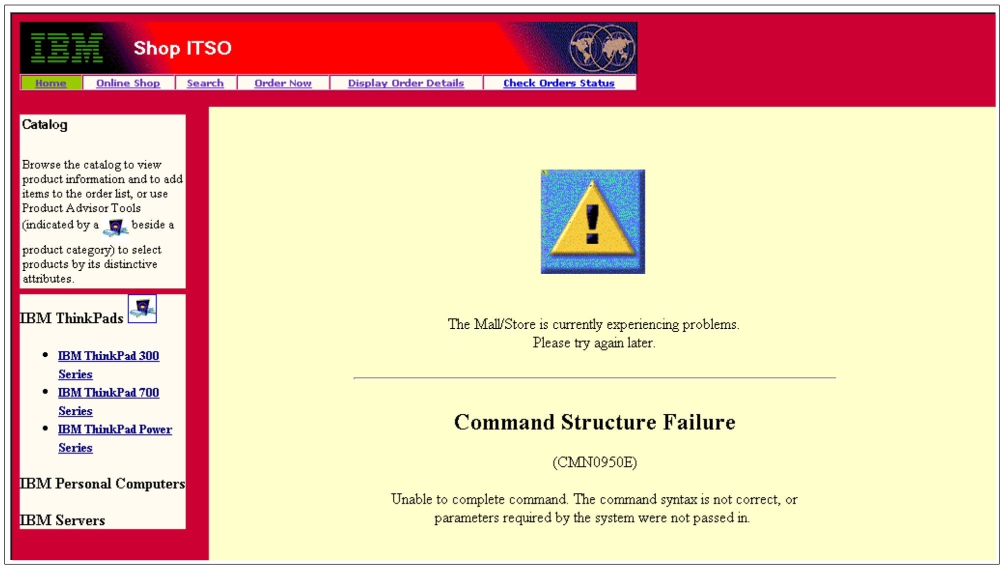
Note: The Product Advisor also has static HTML error pages in this directory.

There is also a macro file (tsslfail.d2w), which is classified as a system error because it can occur at anytime when a new user visits the site. It is stored in the QIBM/ProdData/NetCommerce/macro/MRIxxxx directory. It indicates that the browser type of the shopper is not listed in the BROWSER table in the database. This means that we do not know if a browser supports SSL, tables, or frames. The macro allows the user to attempt to connect through SSL. If it is successful, the BROWSER table is updated.

The system error pages are static HTML files. All pages are global for the mall. You can customize the error pages. For example, you can change graphics and text to your mall style, or add links to your home page or e-mail support.

Consider that you have no variables to indicate which specific error occurred or what the command parameters were. Therefore, you do not have the URL of the original request available to retry.

Your error page must be general purpose so that it caters to all circumstances that generate that error and allows the user to respond accordingly. Figure 228 on page 274 shows one of the original system error pages.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

IBM ThinkPads

- IBM ThinkPad 300 Series
- IBM ThinkPad 700 Series
- IBM ThinkPad Power Series

IBM Personal Computers

IBM Servers

The Mall/Store is currently experiencing problems.
Please try again later.

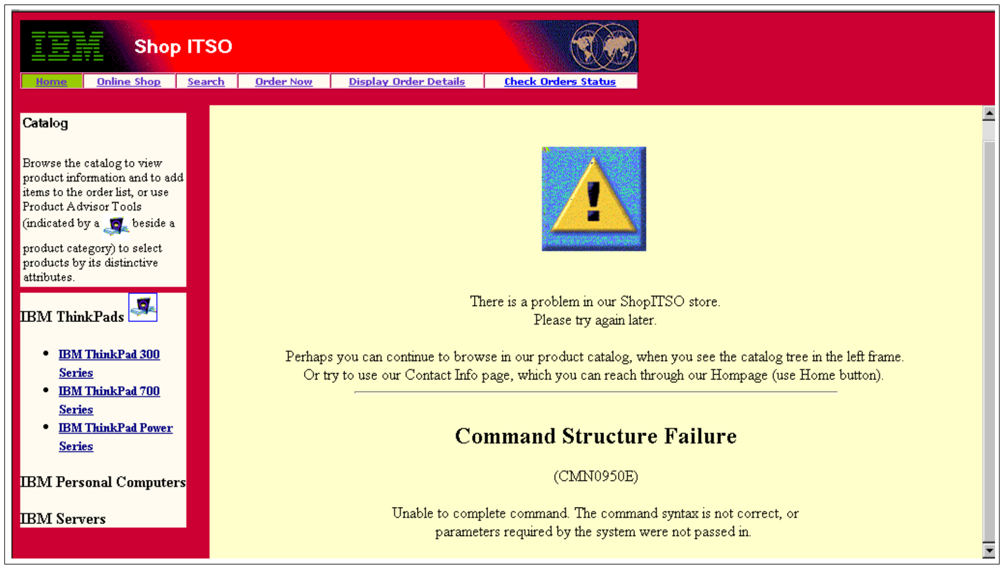
Command Structure Failure

(CMN0950E)

Unable to complete command. The command syntax is not correct, or parameters required by the system were not passed in.

Figure 228. Original System Error Page

We changed the cmdinc.html file. See A.8.13, "CMDINC HTML" on page 479, for more information. The new content of this page is shown in Figure 228.



Catalog

Browse the catalog to view product information and to add items to the order list, or use Product Advisor Tools (indicated by a beside a product category) to select products by its distinctive attributes.

IBM ThinkPads

- IBM ThinkPad 300 Series
- IBM ThinkPad 700 Series
- IBM ThinkPad Power Series

IBM Personal Computers

IBM Servers

There is a problem in our ShopITSO store.
Please try again later.

Perhaps you can continue to browse in our product catalog, when you see the catalog tree in the left frame.
Or try to use our Contact Info page, which you can reach through our Hompage (use Home button).

Command Structure Failure

(CMN0950E)

Unable to complete command. The command syntax is not correct, or parameters required by the system were not passed in.

Figure 229. Changed System Error Page

Chapter 14. Enhancing the Store Using Product Advisor

This chapter covers the following topics:

- A short description of what is Product Advisor
- The approach we used to enhance our sample store using Product Advisor
- How we used Template Designer to customize Product Advisor pages for our sample store
- How we implemented Product Advisor metaphors for our sample store
- The approach we used for publishing the Product Advisor metaphors in the category pages

In this chapter, we primarily document what we did to implement Product Advisor in our sample store. You may need to perform additional steps in your environment.

14.1 What a Product Advisor Is

Product Advisor is an application that is integrated into Net.Commerce V3.2 to enhance the navigation of the product catalog. It allows you to build your own product expertise into the navigational structure of the catalog by creating intelligent shopping metaphors. The metaphors mimic real-life shopping activities such as answering the questions of a salesperson and comparing similar products.

Product Advisor works best where there are similar products with multiple attributes in the Net.Commerce database. It builds its own database tables to specify how the metaphors will be presented. Before you can decide to use Product Advisor and build metaphors, you must make sure your category and product data is suitable for use with Product Advisor. You can only use Product Advisor effectively if:

- Similar Products that can be compared with eachother appear in the same category.
- The features that will be used to make the comparison were entered as product or item attributes in the Net.Commerce database.

If you use the Sales Assistance metaphor, you need to capture the product expertise of your sales force as questions and answers. These questions and answers depend on the attributes that you create.

Once the data and metaphors are designed, you need to create the data in the Net.Commerce database.

Product Advisor consists of a set of creation tools that are Java applets that run in a browser. They store the structure of the metaphor data in the Net.Commerce database. You also need to use the Template Designer to create HTML files for the metaphor pages. You embed special tags in these files to specify where the metaphor content is placed.

The creation tools of Product Advisor are Java applets that run in a browser. These tools include:

Catalog Builder	Determines the data types and sizes of the product attributes (features) that will be used in shopping metaphors.
Metaphor Builder	Determines how the features will be displayed in a metaphor and how different metaphors link together.
Template Designer	Creates HTML template files containing special Product Advisor tags for the metaphor pages.

The run-time components of Product Advisor are Java servlets that run as part of the Web server. *Metaphor Viewers* use the metaphor data in the Net.Commerce database and the template files to build the metaphor views.

14.1.1 Catalog Builder

You use the Catalog Builder to select the product features that will be displayed and to indicate how this information will be used by the metaphors. The shopping metaphor builders define how the catalog will be presented to shoppers.

The catalog keeps track of how many products are offered in each product category. If a category contains subcategories, the count indicates the total number of products in all subcategories.

The Catalog Builder does not add, change, or delete information about your products. It allows you to choose which data to display in the catalog and how you want it displayed. You must run Catalog Builder before you can build the metaphors for a category.

14.1.1.1 Shopping Metaphors

People shop in different ways depending on their needs and their product knowledge. The Product Advisor organizes and presents your catalog data in different ways, so that shoppers can search for products in the manner that is best for them. The different styles of presentation are called shopping metaphors, and the tools that you use to set up the metaphors for each category of products are called *shopping metaphor builders*. For each metaphor, you use the corresponding metaphor builder to define settings that determine which data from the catalog will be presented and how it will be presented. You also use it to specify a template, which determines the overall appearance of the Web pages that the shopper will see. If you set up the templates with metaphor links, the shopper can switch to a different metaphor for the same category at anytime.

There are three shopping metaphors in the Product Advisor:

Sales assistance

A shopper with no technical product knowledge that usually asks questions of a sales assistant in a real store. You can build sample questions that choose particular products or attribute combinations based on the answers given by the shopper. This allows you to build your product knowledge into the catalog and make the shopper's decision process easier.

Product exploration

A shopper who is familiar with a family of similar products may look for particular features. When you have many products in a category with a number of similar attributes (for example, size, color, and material), Product Exploration allows the shopper to look for all products with one attribute and then all remaining products with another attribute and so on until the shopper narrows the list down.

Product comparison
If there are multiple similar products with a few attributes that differentiate them, the shopper can compare them side-by-side to help make a decision.

14.2 Enhancing Our Sample Store Using Product Advisor

To enhance our sample store, we used all the Product Advisor metaphors:

- Product Exploration metaphor
- Product Comparison metaphor
- Sales Assistant metaphor

Because our sample store only has products in the IBM ThinkPads category, we implemented the Product Advisor catalog only for that category.

14.3 Implementing Product Advisor Metaphors

There are several steps required to implement the Product Advisor metaphors. They include:

1. Loading the Product Advisor Applet
2. Running the Catalog Builder
3. Running the Product Exploration Builder
4. Running the Product Comparison Builder
5. Running the Sales Assistant Builder

The first two steps must be performed in order. The remaining steps are optional and only required if you want to use the metaphor associated with the step. For example, if you want to support only the Product Comparison metaphor you would perform steps one, two, and four.

14.3.1 Loading the Product Advisor Applet

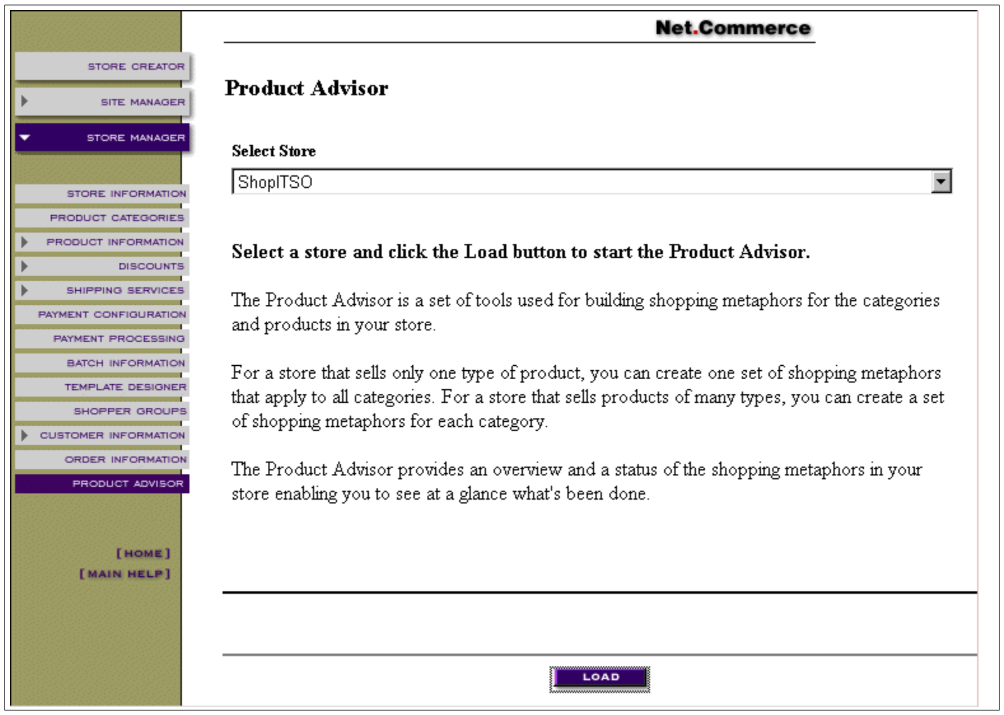
The first step is to load the Product Advisor Applet to the administration PC. This is a Java applet that is downloaded as needed. The length of time required to perform this task is based on the network throughput.

To start the Product Advisor applet, complete these steps:

1. Login as `ncadmin` in the Net.Commerce Administrator Web page at the following URL:

   http://*fullyqualifiedhost_name*/ncadmin

2. From the Product Advisor page in the Store Manager view, shown in Figure 230 on page 278, select the **ShopITSO** store. Click the **Load** button to load the Product Advisor applet.



Product Advisor

Select Store

ShopITSO

Select a store and click the Load button to start the Product Advisor.

The Product Advisor is a set of tools used for building shopping metaphors for the categories and products in your store.

For a store that sells only one type of product, you can create one set of shopping metaphors that apply to all categories. For a store that sells products of many types, you can create a set of shopping metaphors for each category.

The Product Advisor provides an overview and a status of the shopping metaphors in your store enabling you to see at a glance what's been done.

Figure 230. Product Advisor Page in the Net.Commerce Administrator

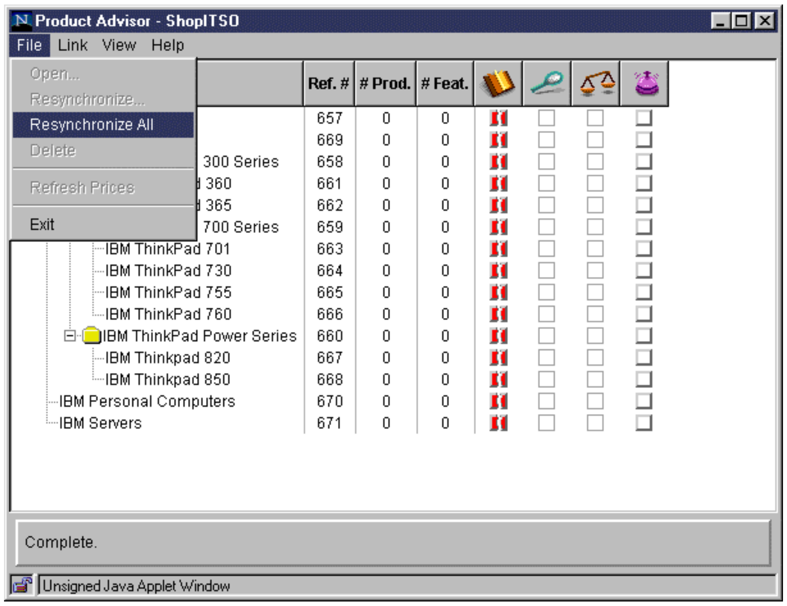
To monitor the JDBC server process that communicates with Product Advisor, connect to the AS/400 system and issue the command:

```
WRKACTJOB SBS(QSERVER)
```

```
                              Work with Active Jobs                     AS01
                                                             04/02/99  14:40:44
CPU %:    13.3     Elapsed time:    00:00:20     Active jobs:   400

Type options, press Enter.
  2=Change    3=Hold    4=End    5=Work with    6=Release    7=Display message
  8=Work with spooled files     13=Disconnect ...

Opt  Subsystem/Job  User        Type  CPU %  Function        Status
     QSERVER        QSYS        SBS      .0                   DEQW
       QPWFSERVSD   QUSER       BCH      .0                   SELW
       QPWFSERVSO   QUSER       PJ       .0                   DEQW
       QSERVER      QPGMR       ASJ      .0                   EVTW
       QZDASOINIT   QUSER       PJ      1.1                   DEQW
       QZDASRVSD    QUSER       BCH      .0                   SELW



Parameters or command
===>
F3=Exit   F5=Refresh       F7=Find      F10=Restart statistics
F11=Display elapsed data   F12=Cancel   F23=More options   F24=More keys
```

Figure 231. Work with Active Jobs

3. All categories are marked broken the first time that the Product Advisor is opened. Select **File —> Resynchronize All**, as shown in Figure 232 on page 279, to start resynchronization.



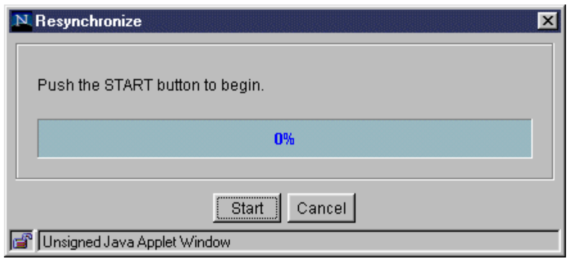
	Ref. #	# Prod.	# Feat.				
	657	0	0		☐	☐	☐
	669	0	0		☐	☐	☐
300 Series	658	0	0		☐	☐	☐
360	661	0	0		☐	☐	☐
365	662	0	0		☐	☐	☐
700 Series	659	0	0		☐	☐	☐
IBM ThinkPad 701	663	0	0		☐	☐	☐
IBM ThinkPad 730	664	0	0		☐	☐	☐
IBM ThinkPad 755	665	0	0		☐	☐	☐
IBM ThinkPad 760	666	0	0		☐	☐	☐
IBM ThinkPad Power Series	660	0	0		☐	☐	☐
IBM Thinkpad 820	667	0	0		☐	☐	☐
IBM Thinkpad 850	668	0	0		☐	☐	☐
IBM Personal Computers	670	0	0		☐	☐	☐
IBM Servers	671	0	0		☐	☐	☐

Complete.

Unsigned Java Applet Window

Figure 232. Product Advisor — Starting Resynchronize All

4. From the resynchronize panel, shown in Figure 233, click the **Start** button to begin resynchronization.

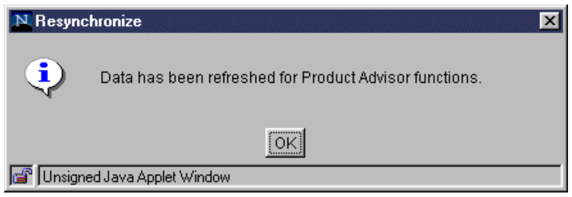


Push the START button to begin.

Unsigned Java Applet Window

Figure 233. Product Advisor — Resynchronization Panel

5. When the data resynchronization finishes successfully, the window shown in Figure 234 pops up. Click **OK** to continue.

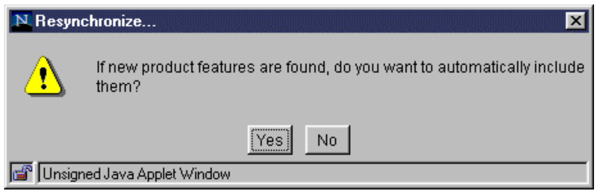


Data has been refreshed for Product Advisor functions.

Unsigned Java Applet Window

Figure 234. Product Advisor — Successful Resynchronization Panel

6. When the window about including new products appears, shown in Figure 235 on page 280, click **Yes** to continue.

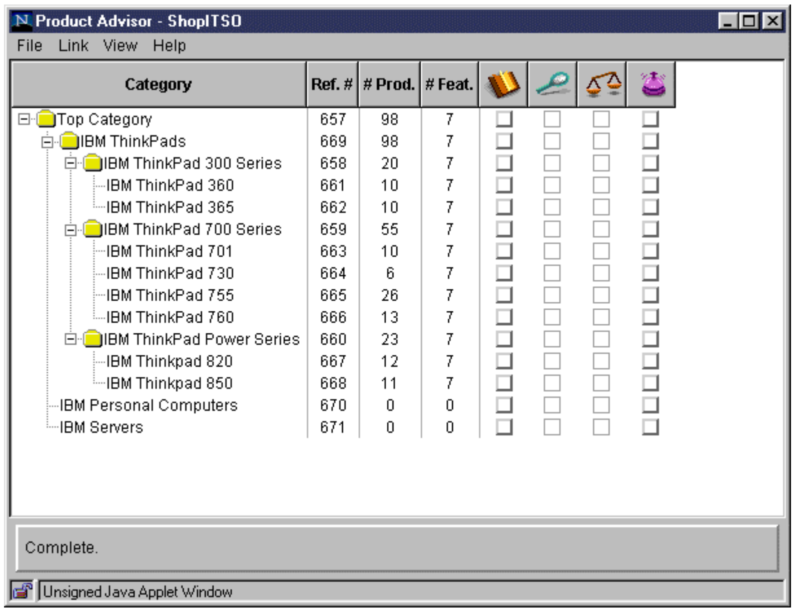


Resynchronize...

If new product features are found, do you want to automatically include them?

Figure 235. Product Advisor — Include New Products Panel

The Product Advisor applet is ready for use when the resynchronization process completes. The window shown in Figure 236 appears.



Category	Ref. #	# Prod.	# Feat.				
Top Category	657	98	7	☐	☐	☐	☐
IBM ThinkPads	669	98	7	☐	☐	☐	☐
IBM ThinkPad 300 Series	658	20	7	☐	☐	☐	☐
IBM ThinkPad 360	661	10	7	☐	☐	☐	☐
IBM ThinkPad 365	662	10	7	☐	☐	☐	☐
IBM ThinkPad 700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Complete.

Figure 236. Product Advisor Applet

14.3.2 Using Catalog Builder

The Catalog Builder allows you to specify which features will be available for use in the metaphors and how they will be used. You *must* run the Catalog Builder *before* you can build the metaphors for a category. Product Advisor uses some fields from standard Net.Commerce tables:

- **PRNBR** — SKU Number
- **PRSDESC** — Product Name
- **PRTHMB** — Thumbnail Image
- **PRURL** — URL
- **PPPRC** — Price

All user-defined attributes from the PRODATR table are also included. For each feature, you can set:

- **Feature Name** —The name that will be displayed in the metaphors.
- **Field Size** — The maximum input field size used in HTML fields such as selection lists.
- **Include** — Whether the feature will be included in metaphors.

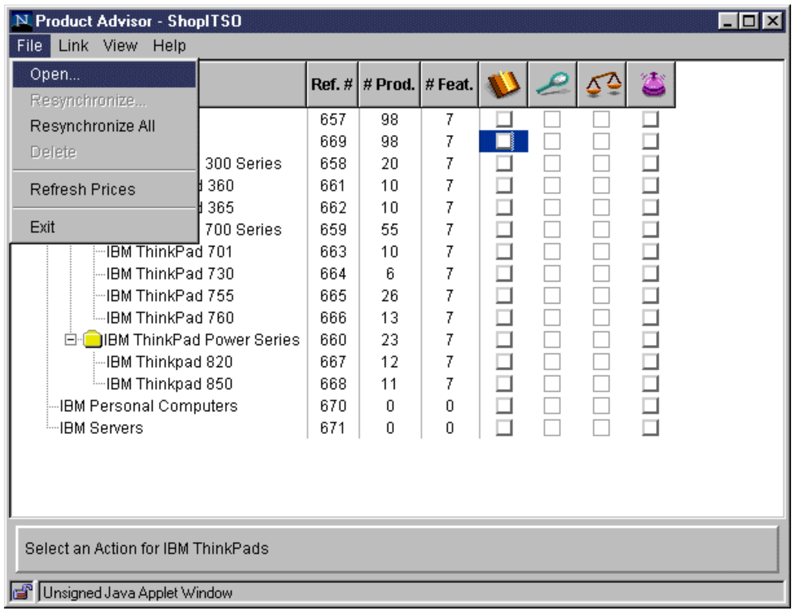
- **Type** (Character, Integer, Decimal,...) — Determines what operations will be allowed on the feature values and how they will be formatted.
- **Usage** (Data, Image, URL, Price) — How the feature values will be used in the HTML output. For example, Data is displayed "as is", but Image will be used in a <IMG> tag.
- **Unit** — Unit of measure that will be appended every time the value is displayed (for example, MB or In).

The Catalog Builder *must* be run *before* you build the metaphors. The exception is Sales Assistant, which allows you to build some questions and answers that do not use attribute values.

When creating a catalog, there are many options that can be done or changed. For our example, we chose not to include thumbnail images. This was done to keep the example simple and to make a change that is easily noticed. By removing the thumbnails, we decreased the amount of time it takes to serve the pages because the image files are not included.

Complete the following steps to run Catalog Builder:

1. From the Product Advisor window, select the **Catalog Builder** icon for the IBM ThinkPads category, as shown in Figure 237. Select **File —> Open** (or click the folder toolbar button) to open the Catalog Builder applet.



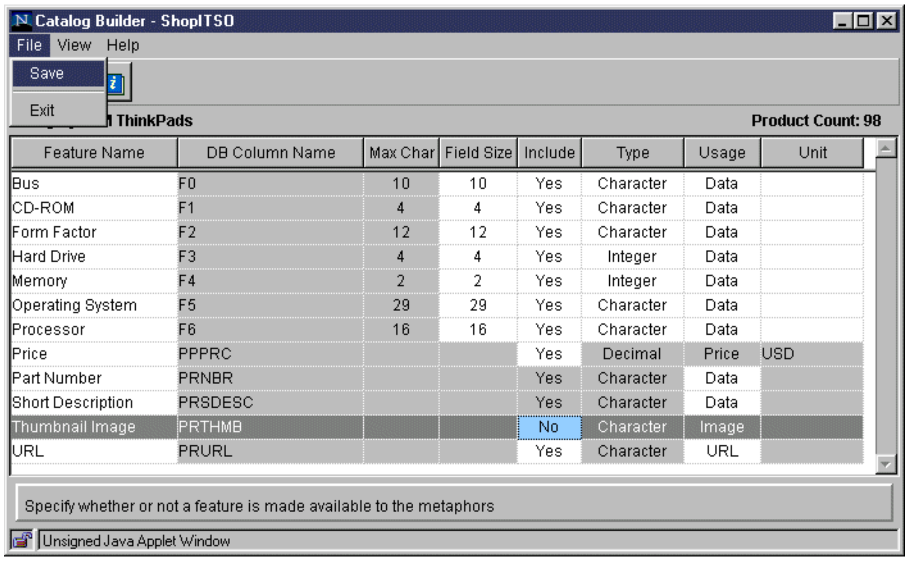
	Ref. #	# Prod.	# Feat.				
	657	98	7	☐	☐	☐	☐
	669	98	7	☐	☐	☐	☐
300 Series	658	20	7	☐	☐	☐	☐
360	661	10	7	☐	☐	☐	☐
365	662	10	7	☐	☐	☐	☐
700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Select an Action for IBM ThinkPads

Unsigned Java Applet Window

Figure 237. Product Advisor — Opening Catalog Builder

2. Double-click on the **Include** field of the Thumbnail Image feature, as shown in Figure 238 on page 282, to change its value from "Yes" to "No". Select **File —> Save** (or click the diskette toolbar button) to save the changes.



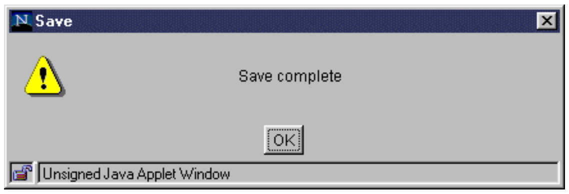
Feature Name	DB Column Name	Max Char	Field Size	Include	Type	Usage	Unit
Bus	F0	10	10	Yes	Character	Data	
CD-ROM	F1	4	4	Yes	Character	Data	
Form Factor	F2	12	12	Yes	Character	Data	
Hard Drive	F3	4	4	Yes	Integer	Data	
Memory	F4	2	2	Yes	Integer	Data	
Operating System	F5	29	29	Yes	Character	Data	
Processor	F6	16	16	Yes	Character	Data	
Price	PPPRC			Yes	Decimal	Price	USD
Part Number	PRNBR			Yes	Character	Data	
Short Description	PRSDESC			Yes	Character	Data	
Thumbnail Image	PRTHMB			No	Character	Image	
URL	PRURL			Yes	Character	URL	

Specify whether or not a feature is made available to the metaphors

Unsigned Java Applet Window

Figure 238. Catalog Builder — Saving Changes

3. The Save Complete panel shown in Figure 239 pops up when the save process is finished. Click **OK** to continue.

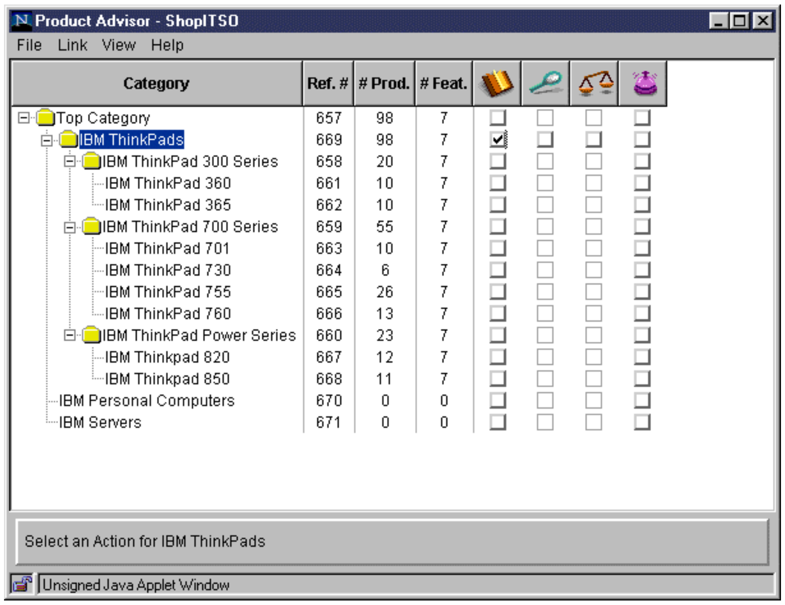


Save complete

Unsigned Java Applet Window

Figure 239. Catalog Builder — Save Complete Panel

4. Select **File —> Exit** (or click the X in the top corner of the window) to exit the Catalog Builder.
5. In the Product Advisor window, the status icon for the IBM ThinkPads category is changed from Enabled (empty box) to **Prepared** (selected box), as shown in Figure 240 on page 283.



Category	Ref. #	# Prod.	# Feat.				
Top Category	657	98	7	☐	☐	☐	☐
IBM ThinkPads	669	98	7	☑	☐	☐	☐
IBM ThinkPad 300 Series	658	20	7	☐	☐	☐	☐
IBM ThinkPad 360	661	10	7	☐	☐	☐	☐
IBM ThinkPad 365	662	10	7	☐	☐	☐	☐
IBM ThinkPad 700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Select an Action for IBM ThinkPads

Unsigned Java Applet Window

Figure 240. Product Advisor — Catalog Prepared

14.3.3 Using Product Exploration Builder

The Product Exploration builder allows you to specify which features will appear in a Product Exploration metaphor, the order in which they will be listed and sorted, and the "widget" used for selection. The widgets relate to HTML form input types. We used:

- **Checkbox** — Useful for textual features with few choices where multiple selections may be used.
- **Single list** — Useful for numeric features with many choices where greater than or less than may be used.
- **Multi list** — Useful for textual features with many choices where multiple selections may be used.

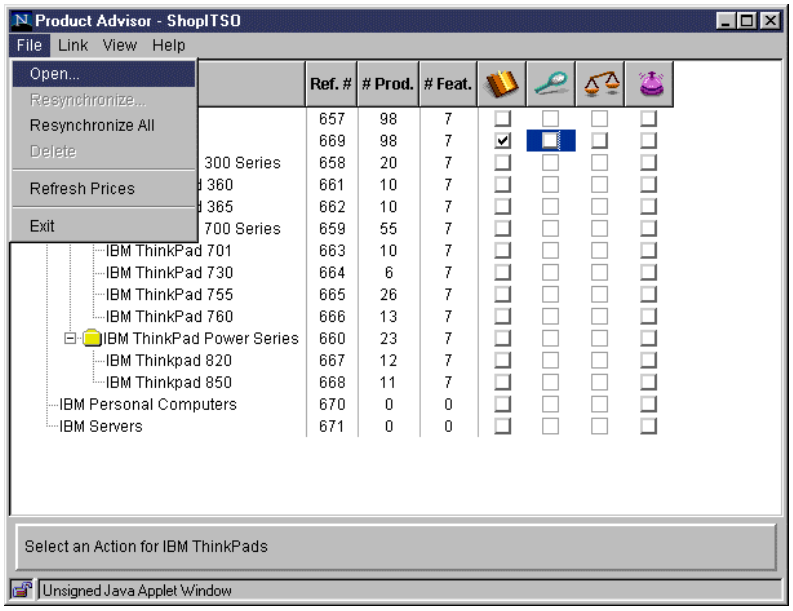
To build a Product Exploration metaphor, you should decide from which of the available features the shopper would want to select and the appropriate widget for the selection. Features that you select, such as Don’t Show, are those that have too many values to narrow the selection (for example, part number or price).

Remember, we are documenting the options we took for our sample. Your options may be different. Select the options that are correct for your environment and design.

Complete the following process to build the Product Exploration metaphor:

1. From the Product Advisor window, select the **Product Exploration Builder** icon for the IBM ThinkPads category, as shown in Figure 241 on page 284. Then, select **File —> Open** to open the Product Exploration Builder applet.

   We chose the IBM ThinkPads category because that was the only category with a catalog created. Notice the check under the book icon. If you click on any category that does not have a catalog created, the Open option is disabled for that selection.



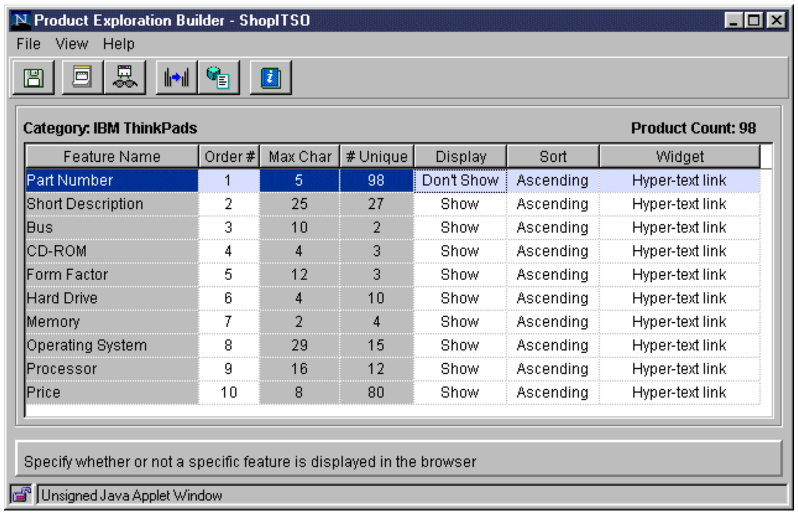
	Ref. #	# Prod.	# Feat.				
	657	98	7	☐	☐	☐	☐
	669	98	7	✓	☐	☐	☐
300 Series	658	20	7	☐	☐	☐	☐
360	661	10	7	☐	☐	☐	☐
365	662	10	7	☐	☐	☐	☐
700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Select an Action for IBM ThinkPads

Unsigned Java Applet Window

Figure 241. Product Advisor — Opening Product Exploration Builder

2. Double-click on the **Display** field of the Part Number feature, as shown in Figure 242, to change its value from "Show" to "Don't Show". Remember this is just *our* option. *Your* options may be different. Select the options that are correct for your environment and design.



Category: IBM ThinkPads

Product Count: 98

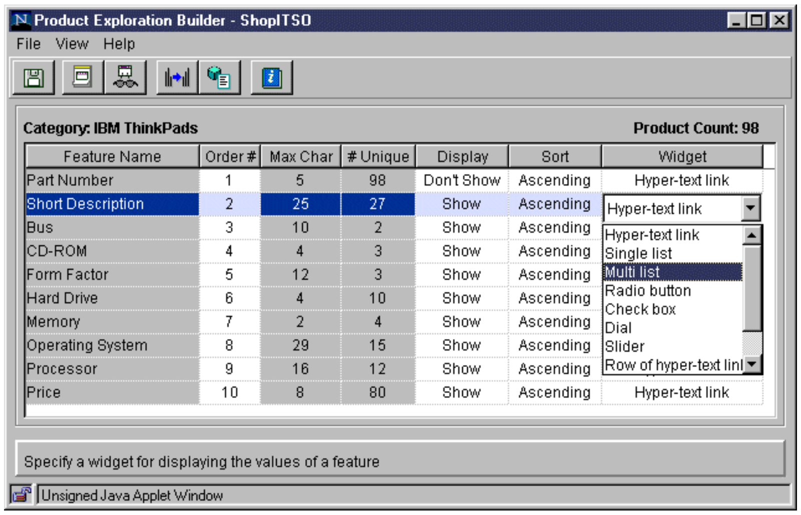
Feature Name	Order #	Max Char	# Unique	Display	Sort	Widget
Part Number	1	5	98	Don't Show	Ascending	Hyper-text link
Short Description	2	25	27	Show	Ascending	Hyper-text link
Bus	3	10	2	Show	Ascending	Hyper-text link
CD-ROM	4	4	3	Show	Ascending	Hyper-text link
Form Factor	5	12	3	Show	Ascending	Hyper-text link
Hard Drive	6	4	10	Show	Ascending	Hyper-text link
Memory	7	2	4	Show	Ascending	Hyper-text link
Operating System	8	29	15	Show	Ascending	Hyper-text link
Processor	9	16	12	Show	Ascending	Hyper-text link
Price	10	8	80	Show	Ascending	Hyper-text link

Specify whether or not a specific feature is displayed in the browser

Unsigned Java Applet Window

Figure 242. Product Exploration Builder — Changing Display Value

3. Click on the **Widget** field of the Short Description feature, as shown in Figure 243 on page 285, to change its value from "Hyper-text link" to "Multi list".


Product Exploration Builder - ShopITSO

Category: IBM ThinkPads **Product Count: 98**

Feature Name	Order #	Max Char	# Unique	Display	Sort	Widget
Part Number	1	5	98	Don't Show	Ascending	Hyper-text link
Short Description	2	25	27	Show	Ascending	Hyper-text link
Bus	3	10	2	Show	Ascending	
CD-ROM	4	4	3	Show	Ascending	
Form Factor	5	12	3	Show	Ascending	
Hard Drive	6	4	10	Show	Ascending	
Memory	7	2	4	Show	Ascending	
Operating System	8	29	15	Show	Ascending	
Processor	9	16	12	Show	Ascending	
Price	10	8	80	Show	Ascending	Hyper-text link

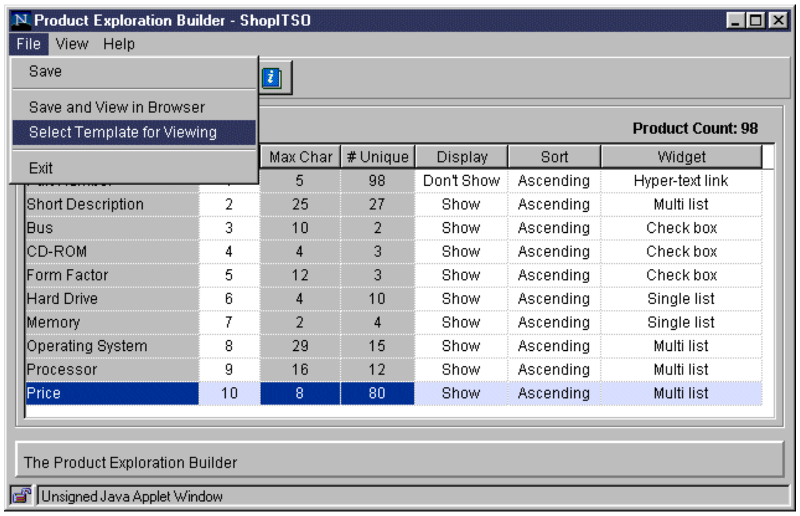
Specify a widget for displaying the values of a feature

Unsigned Java Applet Window

Figure 243. Product Exploration Builder — Changing Widget Value

4. Repeat step three for the rest of the features, as shown in Figure 244, by using the following values:
   - **Bus** — Check box
   - **CD-ROM** — Check box
   - **Form Factor** — Check box
   - **Hard Drive** — Single list
   - **Memory** — Single list
   - **Operating System** — Multi list
   - **Processor** — Multi list
   - **Price** — Multi list

   Then, select **File —> Select Template for Viewing** to select the template for viewing the Product Exploration page.


Product Exploration Builder - ShopITSO

Product Count: 98

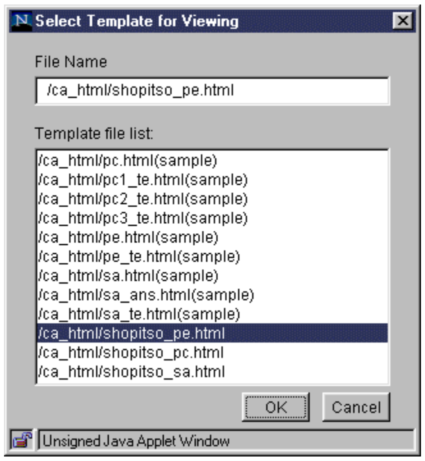
		Max Char	# Unique	Display	Sort	Widget
		5	98	Don't Show	Ascending	Hyper-text link
Short Description	2	25	27	Show	Ascending	Multi list
Bus	3	10	2	Show	Ascending	Check box
CD-ROM	4	4	3	Show	Ascending	Check box
Form Factor	5	12	3	Show	Ascending	Check box
Hard Drive	6	4	10	Show	Ascending	Single list
Memory	7	2	4	Show	Ascending	Single list
Operating System	8	29	15	Show	Ascending	Multi list
Processor	9	16	12	Show	Ascending	Multi list
Price	10	8	80	Show	Ascending	Multi list

The Product Exploration Builder

Unsigned Java Applet Window

Figure 244. Product Exploration Builder — Selecting Template for Viewing

5. From the Select Template for Viewing panel, shown in Figure 245 on page 286, select **/ca_html/shopitso_pe.html** (created in 14.4.2, “Customizing the Product Exploration Page” on page 302). Click **OK** to continue.



Select Template for Viewing

File Name

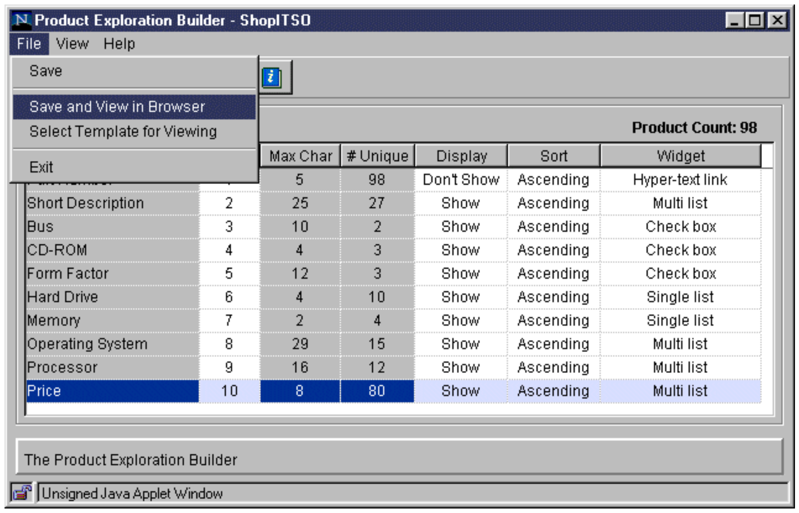
/ca_html/shopitso_pe.html

Template file list:

/ca_html/pc.html(sample)
/ca_html/pc1_te.html(sample)
/ca_html/pc2_te.html(sample)
/ca_html/pc3_te.html(sample)
/ca_html/pe.html(sample)
/ca_html/pe_te.html(sample)
/ca_html/sa.html(sample)
/ca_html/sa_ans.html(sample)
/ca_html/sa_te.html(sample)
/ca_html/shopitso_pe.html
/ca_html/shopitso_pc.html
/ca_html/shopitso_sa.html

Figure 245. Product Exploration Builder — Select Template for Viewing Panel

6. Select **File —> Save and View in Browser** (or click the glasses toolbar button), as shown in Figure 246, to save and view the Product Exploration page in a browser.



Product Exploration Builder - ShopITSO

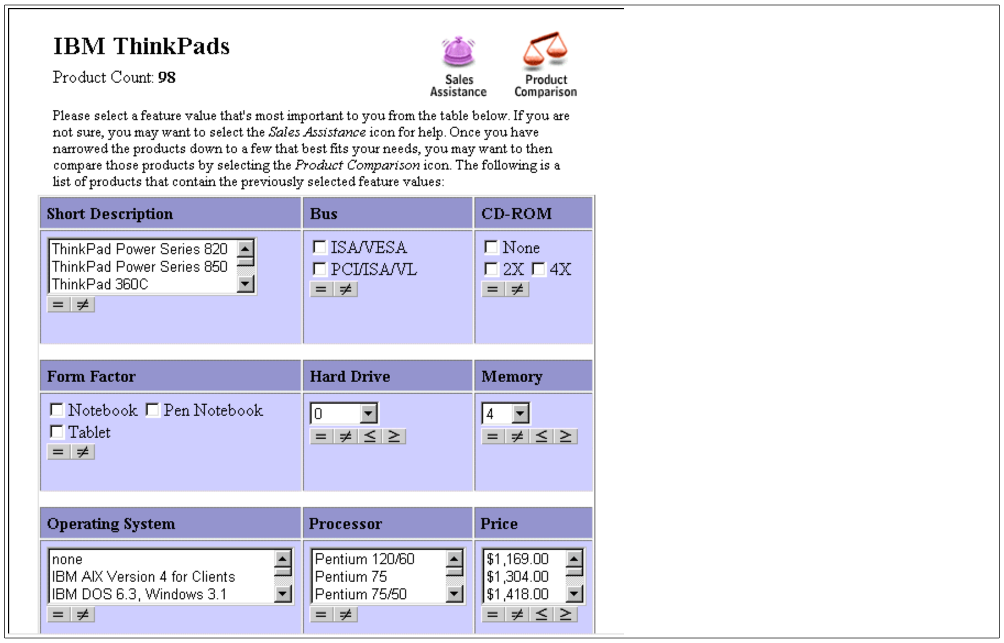
Product Count: 98

		Max Char	# Unique	Display	Sort	Widget
[illegible]	[illegible]	5	98	Don't Show	Ascending	Hyper-text link
Short Description	2	25	27	Show	Ascending	Multi list
Bus	3	10	2	Show	Ascending	Check box
CD-ROM	4	4	3	Show	Ascending	Check box
Form Factor	5	12	3	Show	Ascending	Check box
Hard Drive	6	4	10	Show	Ascending	Single list
Memory	7	2	4	Show	Ascending	Single list
Operating System	8	29	15	Show	Ascending	Multi list
Processor	9	16	12	Show	Ascending	Multi list
Price	10	8	80	Show	Ascending	Multi list

The Product Exploration Builder

Figure 246. Product Exploration Builder — Viewing Product Exploration Page

7. The Product Exploration page, as shown in Figure 247, pops up in a browser. Close the browser to continue.


IBM ThinkPads

Product Count: **98**

Please select a feature value that's most important to you from the table below. If you are not sure, you may want to select the *Sales Assistance* icon for help. Once you have narrowed the products down to a few that best fits your needs, you may want to then compare those products by selecting the *Product Comparison* icon. The following is a list of products that contain the previously selected feature values:

Short Description	Bus	CD-ROM
ThinkPad Power Series 820 ThinkPad Power Series 850 ThinkPad 360C = ≠	☐ ISA/VESA ☐ PCI/ISA/VL = ≠	☐ None ☐ 2X ☐ 4X = ≠
Form Factor	**Hard Drive**	**Memory**
☐ Notebook ☐ Pen Notebook ☐ Tablet = ≠	0 = ≠ ≤ ≥	4 = ≠ ≤ ≥
Operating System	**Processor**	**Price**
none IBM AIX Version 4 for Clients IBM DOS 6.3, Windows 3.1 = ≠	Pentium 120/60 Pentium 75 Pentium 75/50 = ≠	$1,169.00 $1,304.00 $1,418.00 = ≠ ≤ ≥

Figure 247. Product Exploration Page

8. Select **File —> Exit** (or click the X in the top corner of the window) to exit the Product Exploration Builder and return to the Product Advisor.

14.3.4 Using Product Comparison Builder

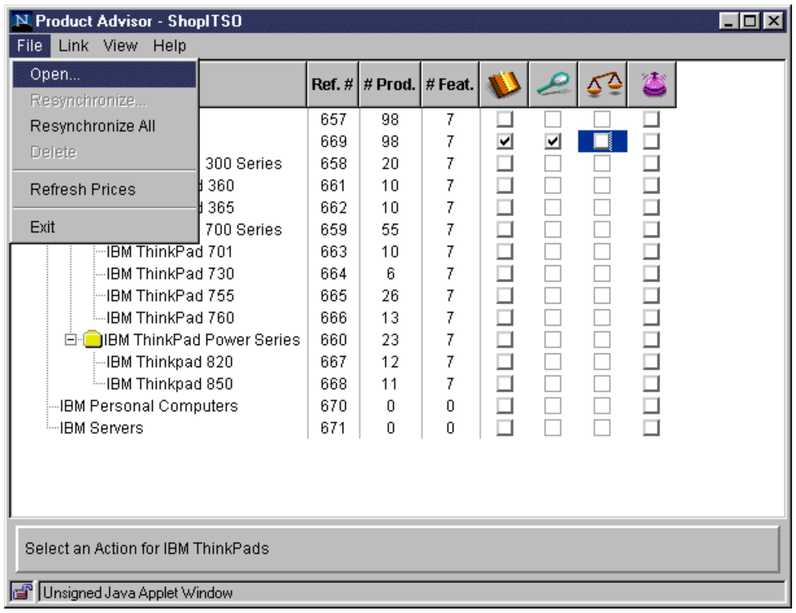
The Product Comparison builder allows you to specify which features will appear in a Product Comparison metaphor, the order in which they will be listed, and which features will link to the product pages. You should specify "Don't Show" for features that cannot usefully be compared (for example, picture or part number). You must make sure one feature links to product pages.

Remember, we are documenting the options we took for *our* sample. *Your* options may be different. Select the options that are correct for your environment and design.

Complete the following steps to build the Product Comparison metaphor:

1. From the Product Advisor window, select the **Product Comparison Builder** icon for the IBM ThinkPads category, as shown in Figure 248 on page 288. Then, select **File —> Open** (or click the folder toolbar button) to open the Product Comparison Builder applet.

   We chose the IBM ThinkPads category because that was the only category with a catalog created (notice the check under the book icon). If you click on any category that does not have a catalog created, the Open option will be disabled for that selection.


Product Advisor - ShopITSO

File Link View Help

Open...
Resynchronize...
Resynchronize All
Delete
Refresh Prices
Exit

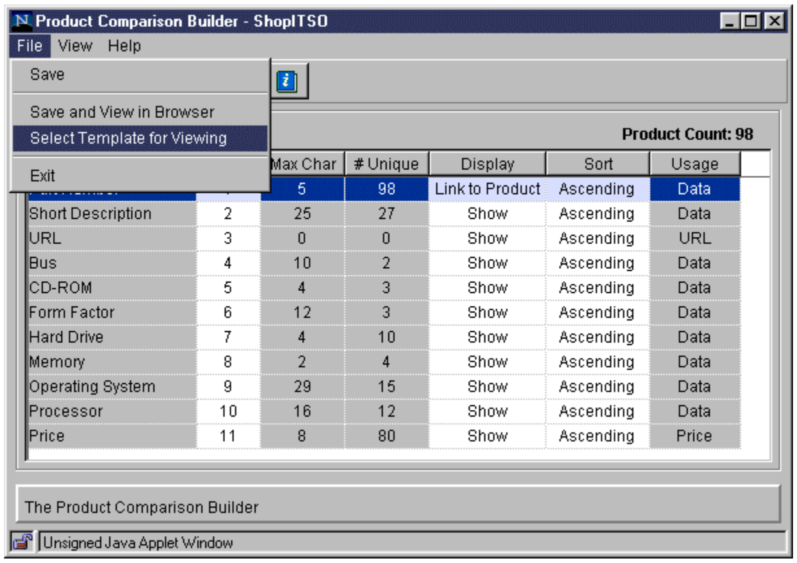
	Ref. #	# Prod.	# Feat.				
	657	98	7	☐	☐	☐	☐
	669	98	7	✓	✓	☐	☐
300 Series	658	20	7	☐	☐	☐	☐
360	661	10	7	☐	☐	☐	☐
365	662	10	7	☐	☐	☐	☐
700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Select an Action for IBM ThinkPads

Unsigned Java Applet Window

Figure 248. Product Advisor — Opening Product Comparison Builder

2. Select **File —> Select Template for Viewing**, as shown in Figure 249, to select the template for viewing the Product Comparison page.


Product Comparison Builder - ShopITSO

File View Help

Save
Save and View in Browser
Select Template for Viewing
Exit

Product Count: 98

		Max Char	# Unique	Display	Sort	Usage
		5	98	Link to Product	Ascending	Data
Short Description	2	25	27	Show	Ascending	Data
URL	3	0	0	Show	Ascending	URL
Bus	4	10	2	Show	Ascending	Data
CD-ROM	5	4	3	Show	Ascending	Data
Form Factor	6	12	3	Show	Ascending	Data
Hard Drive	7	4	10	Show	Ascending	Data
Memory	8	2	4	Show	Ascending	Data
Operating System	9	29	15	Show	Ascending	Data
Processor	10	16	12	Show	Ascending	Data
Price	11	8	80	Show	Ascending	Price

The Product Comparison Builder

Unsigned Java Applet Window

Figure 249. Product Comparison Builder — Selecting Template for Viewing

3. From the Select Template for Viewing panel, shown in Figure 250, select **/ca_html/shopitso_pc.html** (created in 14.4.3, “Customizing the Product Comparison Page” on page 307). Click **OK** to continue.

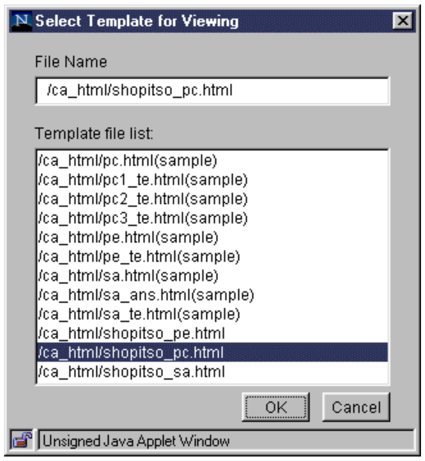
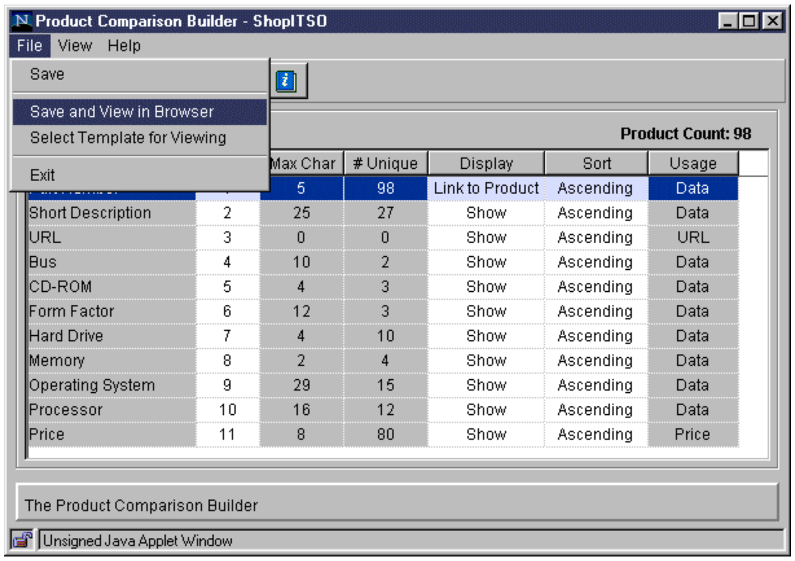

Select Template for Viewing
File Name
/ca_html/shopitso_pc.html
Template file list:
/ca_html/pc.html(sample)
/ca_html/pc1_te.html(sample)
/ca_html/pc2_te.html(sample)
/ca_html/pc3_te.html(sample)
/ca_html/pe.html(sample)
/ca_html/pe_te.html(sample)
/ca_html/sa.html(sample)
/ca_html/sa_ans.html(sample)
/ca_html/sa_te.html(sample)
/ca_html/shopitso_pe.html
/ca_html/shopitso_pc.html
/ca_html/shopitso_sa.html
OK
Cancel
Unsigned Java Applet Window

Figure 250. Product Comparison Builder — Select Template for Viewing Panel

4. Select **File —> Save and View in Browser** (or click the glasses toolbar button) as shown in Figure 251, to save and view the Product Comparison page in a browser.

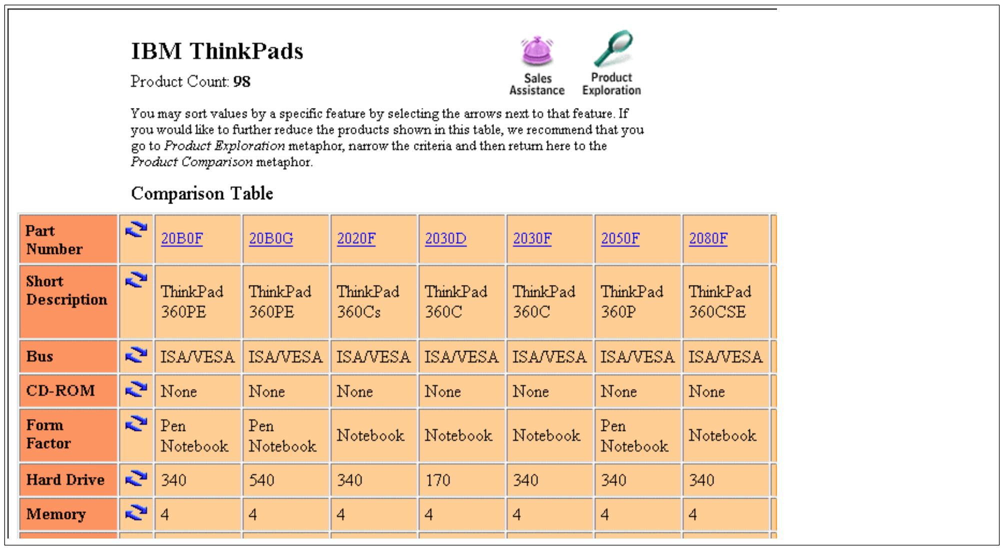

Product Comparison Builder - ShopITSO
File
View
Help
Save
Save and View in Browser
Select Template for Viewing
Exit
Product Count: 98
The Product Comparison Builder
Unsigned Java Applet Window

		Max Char	# Unique	Display	Sort	Usage
		5	98	Link to Product	Ascending	Data
Short Description	2	25	27	Show	Ascending	Data
URL	3	0	0	Show	Ascending	URL
Bus	4	10	2	Show	Ascending	Data
CD-ROM	5	4	3	Show	Ascending	Data
Form Factor	6	12	3	Show	Ascending	Data
Hard Drive	7	4	10	Show	Ascending	Data
Memory	8	2	4	Show	Ascending	Data
Operating System	9	29	15	Show	Ascending	Data
Processor	10	16	12	Show	Ascending	Data
Price	11	8	80	Show	Ascending	Price

Figure 251. Product Comparison Builder — Viewing Product Comparison Page

Refer to C.1.1, "Net.Commerce Online Documentation" on page 529, for details about accessing the documentation.

5. The Product Comparison page, shown in Figure 252 on page 290, pops up in a browser. Notice how the Product Comparison page compares the different products in the table. Close the browser to continue defining the metaphors.


IBM ThinkPads

Product Count: 98

You may sort values by a specific feature by selecting the arrows next to that feature. If you would like to further reduce the products shown in this table, we recommend that you go to *Product Exploration* metaphor, narrow the criteria and then return here to the *Product Comparison* metaphor.

Comparison Table

Part Number	⇄	20B0F	20B0G	2020F	2030D	2030F	2050F	2080F
Short Description	⇄	ThinkPad 360PE	ThinkPad 360PE	ThinkPad 360Cs	ThinkPad 360C	ThinkPad 360C	ThinkPad 360P	ThinkPad 360CSE
Bus	⇄	ISA/VESA	ISA/VESA	ISA/VESA	ISA/VESA	ISA/VESA	ISA/VESA	ISA/VESA
CD-ROM	⇄	None	None	None	None	None	None	None
Form Factor	⇄	Pen Notebook	Pen Notebook	Notebook	Notebook	Notebook	Pen Notebook	Notebook
Hard Drive	⇄	340	540	340	170	340	340	340
Memory	⇄	4	4	4	4	4	4	4

Figure 252. Product Comparison Page

6. Select **File —> Exit** (or click the X in the top corner of the window) to exit the Product Comparison Builder and return to Product Advisor.

14.3.5 Using Sales Assistant Builder

The Sales Assistant builder allows you to build the question and answer trees for the Sales Assistance metaphor. You start by adding a question and then multiple answers under the question. Each answer can have constraints applied and can link to:

- Another question
- Another Sales Assistance metaphor
- Another metaphor (Product Exploration or Product Comparison)
- Another Web page URL

The default link, which is used if there are no further questions or metaphor links, can be changed. The initial setting is the Product Comparison metaphor.

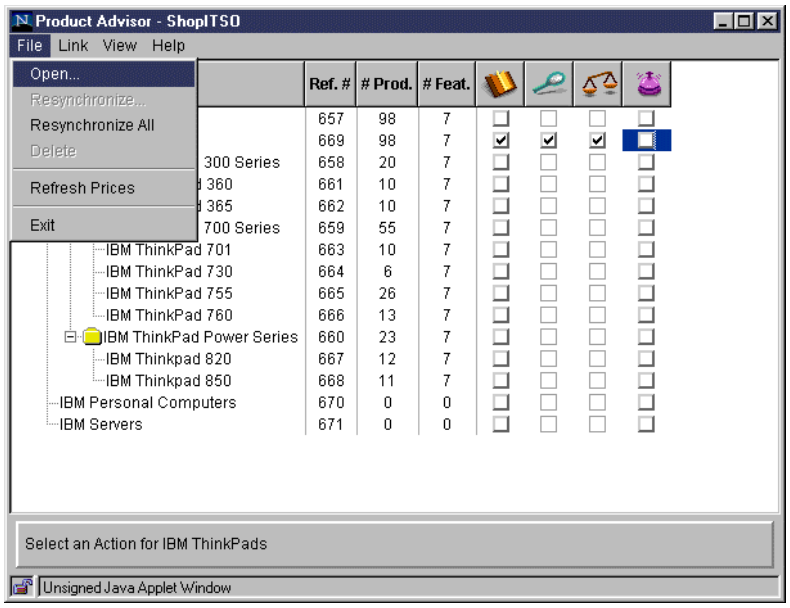
A useful function in Sales Assistant builder is the ability to import the question and answer tree from another category. You can build Sales Assistance metaphors without the catalog data as long as you do not intend to use constraints or other metaphors. For example, you can build a sequence of questions and answers that just link to a particular category or product.

Remember, we are documenting the options we took for *our* sample. We added one question and five answers. In the real world, *you* will have many more questions and answers. Select the options that are correct for your environment and design.

Complete these tasks to build the Sales Assistant metaphor:

1. From the Product Advisor window, select the **Sales Assistant Builder** icon for the IBM ThinkPads category, as shown in Figure 253 on page 291. Then, select **File —> Open** (or click the folder toolbar button) to open the Sales Assistant Builder applet.

We chose the IBM ThinkPads category because that was the only category with a catalog created (notice the check under the book icon). If you click on any category that does not have a catalog created, the Open option will be disabled for that selection.



Product Advisor - ShopITSO

File Link View Help

Open...
Resynchronize...
Resynchronize All
Delete
Refresh Prices
Exit

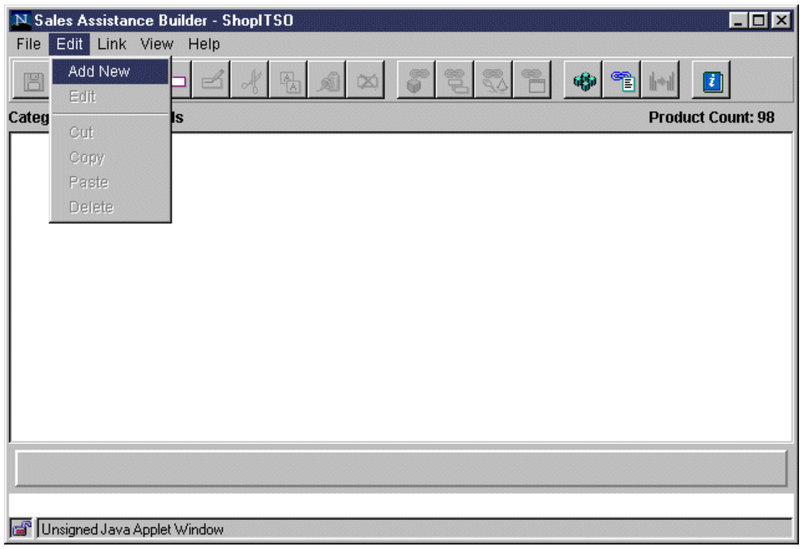
	Ref. #	# Prod.	# Feat.				
	657	98	7	☐	☐	☐	☐
	669	98	7	✓	✓	✓	☐
300 Series	658	20	7	☐	☐	☐	☐
360	661	10	7	☐	☐	☐	☐
365	662	10	7	☐	☐	☐	☐
700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Select an Action for IBM ThinkPads

Unsigned Java Applet Window

Figure 253. Product Advisor — Opening Sales Assistant Builder

2. Select **Edit —> Add New** (or click the white rectangular toolbar button), as shown in Figure 254, to add the first question of the Sales Assistant.



Sales Assistance Builder - ShopITSO

File Edit Link View Help

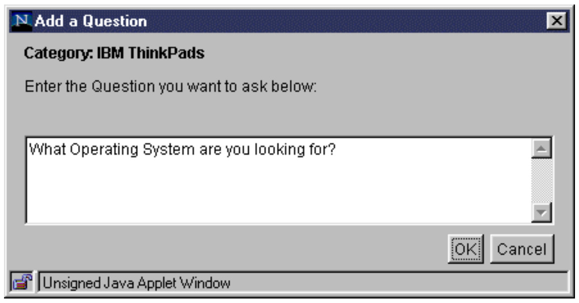
Add New
Edit
Cut
Copy
Paste
Delete

Categ ls Product Count: 98

Unsigned Java Applet Window

Figure 254. Sales Assistant Builder — Adding a Question

3. In the Add a Question panel, type: `What Operating System are you looking for?`, as shown in Figure 255 on page 292. Click the **OK** button to continue.


Add a Question

Category: IBM ThinkPads

Enter the Question you want to ask below:

What Operating System are you looking for?

Figure 255. Sales Assistant Builder — Add a Question Panel

4. Select the question. Then, select **Edit —> Add New** (or click the white rectangle toolbar button), as shown in Figure 256, to add the first answer for the question.

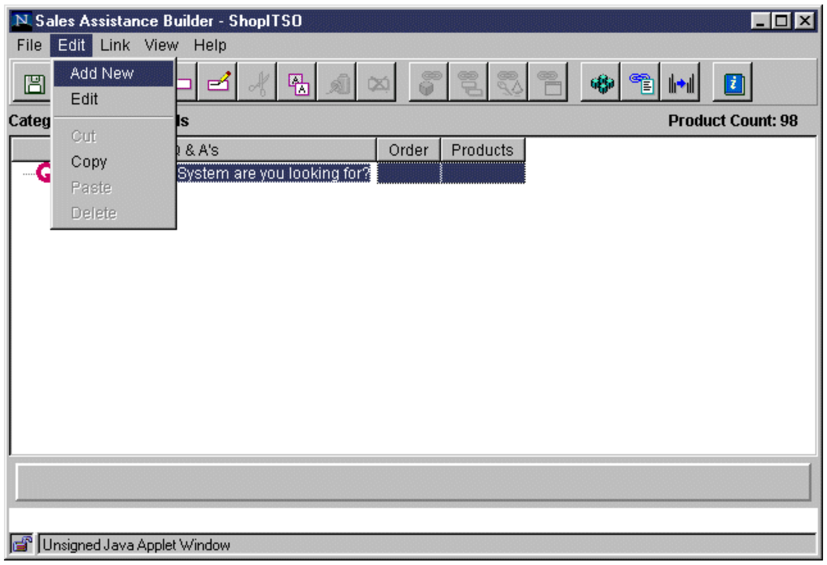
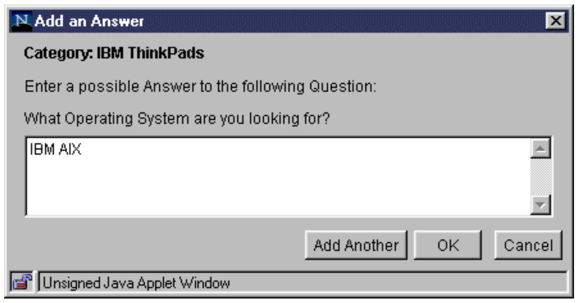

Sales Assistance Builder - ShopITSO
File
Edit
Link
View
Help
Add New
Edit
Cut
Copy
Paste
Delete
Product Count: 98
& A's
Order
Products
System are you looking for?
Unsigned Java Applet Window

Figure 256. Sales Assistant Builder — Adding an Answer

5. Type `IBM AIX` in the Add an Answer panel, as shown in Figure 257 on page 293, and click the **Add Another** button to continue. Add the following answers:

   - `IBM DOS`
   - `MS Windows 95`
   - `MS Windows NT`
   - `Other`

   Click **OK** to continue.


Add an Answer

Category: IBM ThinkPads

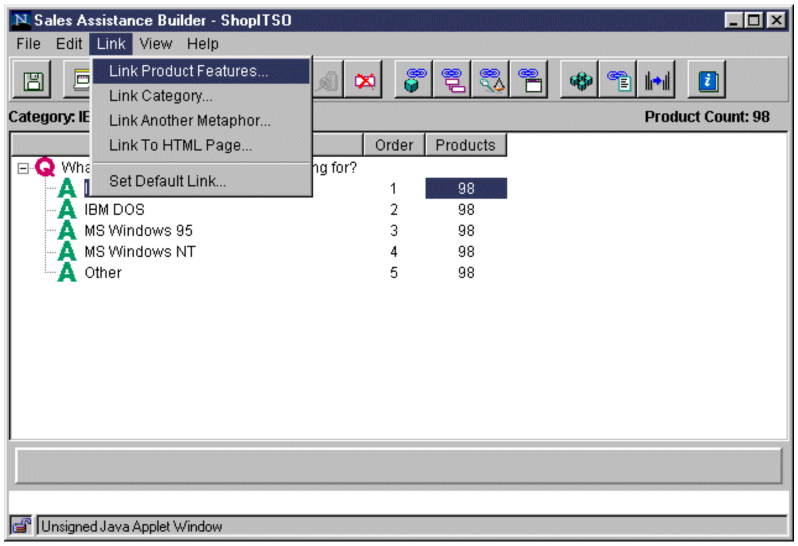
Enter a possible Answer to the following Question:

What Operating System are you looking for?

IBM AIX

Figure 257. Sales Assistant Builder — Add an Answer Panel

6. Select the **IBM AIX** answer. Then, select **Link —> Link Product Features...** (or click the green cube toolbar button), as shown in Figure 258, to select the features for product constrains.



Category: IE

Product Count: 98

	Order	Products
Wha... ng for?		
I	1	98
IBM DOS	2	98
MS Windows 95	3	98
MS Windows NT	4	98
Other	5	98

Figure 258. Sales Assistant Builder — Selecting Product Constrains

7. From the Select products constrains panel shown in Figure 259 on page 294, select the **Operating System** feature. Then, select the **IBM AIX** feature value. Click the **Add >>>** button. Then, click **OK** to continue.

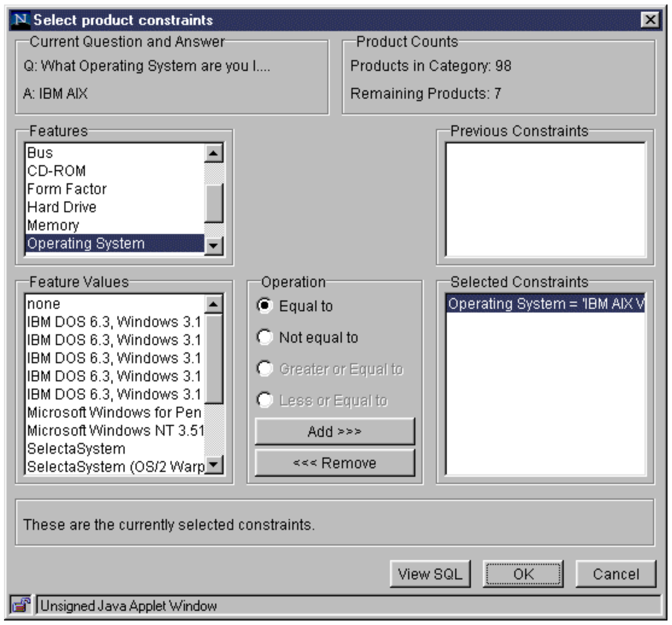


Figure 259. Sales Assistant Builder — Select Product Constrains Panel

8. Select the **IBM AIX** answer. Then, select **Link —> Link Another Metaphor...**, as shown in Figure 260, to change the default link metaphor.

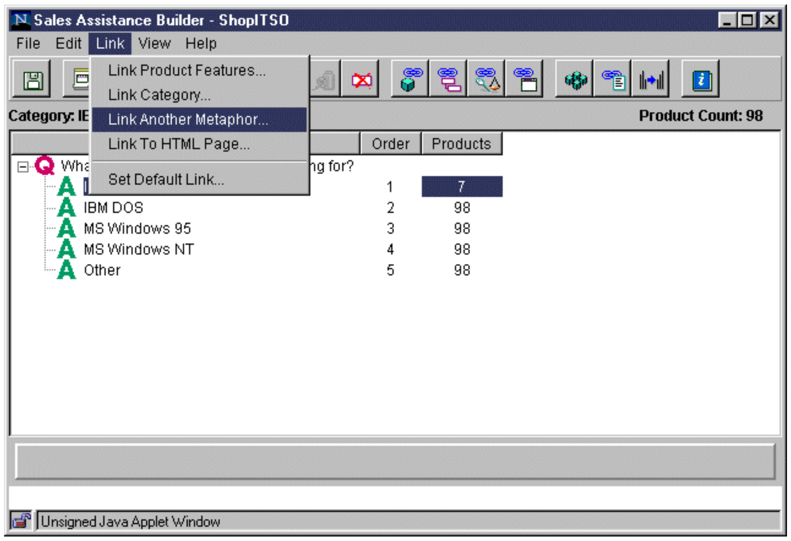


Figure 260. Sales Assistant Builder — Linking to Another Metaphor

9. From the Link to Another Metaphor panel, shown in Figure 261 on page 295, select **Product Exploration/IBM ThinkPads** and click the **List...** button to continue.

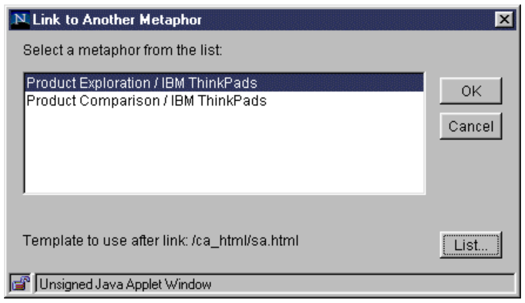

Link to Another Metaphor
Select a metaphor from the list:
Product Exploration / IBM ThinkPads
Product Comparison / IBM ThinkPads
OK
Cancel
Template to use after link: /ca_html/sa.html
List...
Unsigned Java Applet Window

Figure 261. Sales Assistant Builder — Link to Another Metaphor Panel

10.From the Select Template for Viewing panel, as shown in Figure 262, select **/ca_html/shopitso_pe.html** for the template file. Click the **OK** button to continue.

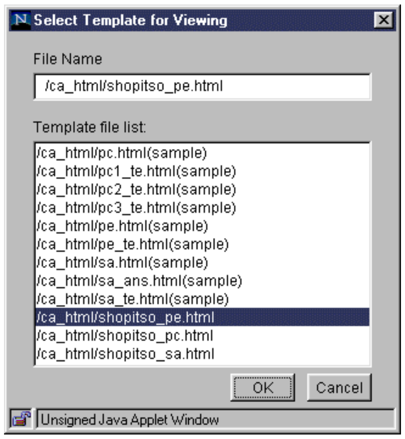
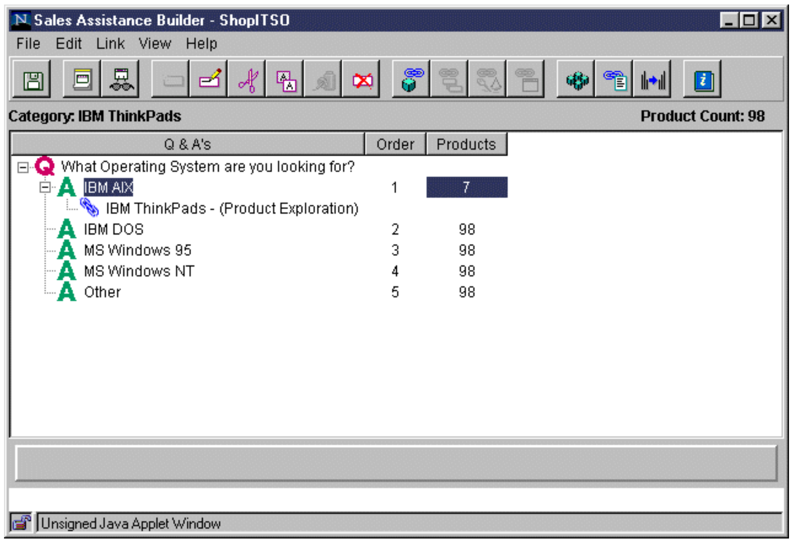

Select Template for Viewing
File Name
/ca_html/shopitso_pe.html
Template file list:
/ca_html/pc.html(sample)
/ca_html/pc1_te.html(sample)
/ca_html/pc2_te.html(sample)
/ca_html/pc3_te.html(sample)
/ca_html/pe.html(sample)
/ca_html/pe_te.html(sample)
/ca_html/sa.html(sample)
/ca_html/sa_ans.html(sample)
/ca_html/sa_te.html(sample)
/ca_html/shopitso_pe.html
/ca_html/shopitso_pc.html
/ca_html/shopitso_sa.html
OK
Cancel
Unsigned Java Applet Window

Figure 262. Sales Assistant Builder — Select Template for Viewing Panel

11.From the Link to Another Metaphor panel, click the **OK** button to continue.

The Sales Assistant Builder, shown in Figure 263 on page 296, now displays the new metaphor link just defined and the new product count constrained from 98 down to 7.


Sales Assistance Builder - ShopITSO
File Edit Link View Help
Unsigned Java Applet Window

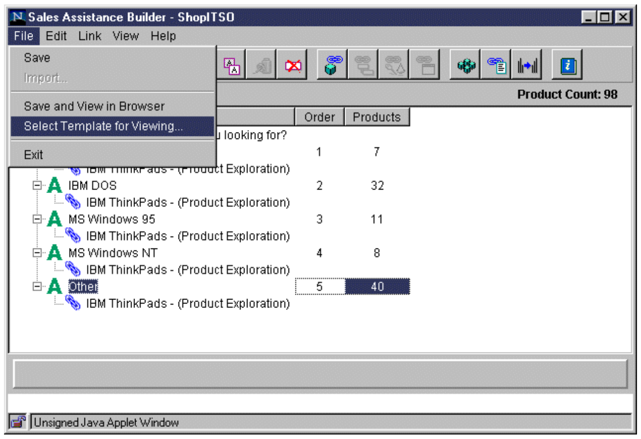
Category: IBM ThinkPads   Product Count: 98

Q & A's	Order	Products
What Operating System are you looking for?		
IBM AIX	1	7
IBM ThinkPads - (Product Exploration)		
IBM DOS	2	98
MS Windows 95	3	98
MS Windows NT	4	98
Other	5	98

Figure 263. Sales Assistant Builder — Link Metaphor and Product Constrains Added

12. Repeat steps six through eleven for the rest of the answers. Constrain the product count for each answer to:
    - IBM DOS: `32`
    - MS Windows 95: `11`
    - MS Windows NT: `8`
    - Other: `40`

13. Select **File —> Select Template for Viewing**, as shown in Figure 264, to select the template for viewing the Sales Assistant page.

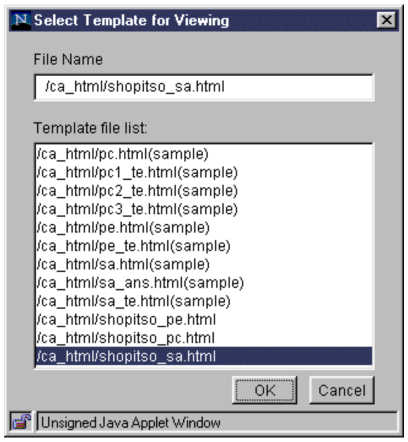

Sales Assistance Builder - ShopITSO
File Edit Link View Help
Save
Import...
Save and View in Browser
Select Template for Viewing...
Exit
Unsigned Java Applet Window

Product Count: 98

	Order	Products
u looking for?		
	1	7
IBM ThinkPads - (Product Exploration)		
IBM DOS	2	32
IBM ThinkPads - (Product Exploration)		
MS Windows 95	3	11
IBM ThinkPads - (Product Exploration)		
MS Windows NT	4	8
IBM ThinkPads - (Product Exploration)		
Other	5	40
IBM ThinkPads - (Product Exploration)		

Figure 264. Sales Assistant Builder — Selecting Template for Viewing

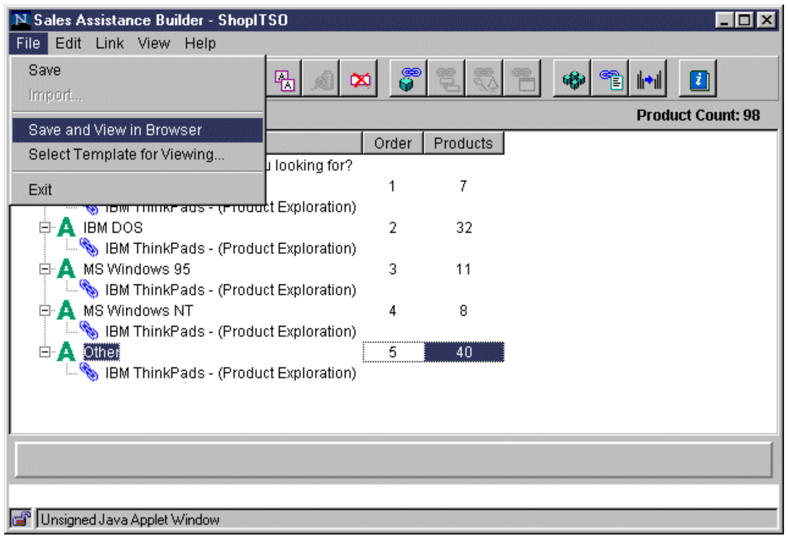
14. From the Select Template for Viewing panel, shown in Figure 265 on page 297, select **/ca_html/shopitso_sa.html** (created in 14.4.4, “Customizing the Sales Assistant Pages” on page 312). Click **OK** to continue.


Select Template for Viewing
File Name
/ca_html/shopitso_sa.html
Template file list:
OK
Cancel
Unsigned Java Applet Window

/ca_html/pc.html(sample)
/ca_html/pc1_te.html(sample)
/ca_html/pc2_te.html(sample)
/ca_html/pc3_te.html(sample)
/ca_html/pe.html(sample)
/ca_html/pe_te.html(sample)
/ca_html/sa.html(sample)
/ca_html/sa_ans.html(sample)
/ca_html/sa_te.html(sample)
/ca_html/shopitso_pe.html
/ca_html/shopitso_pc.html
/ca_html/shopitso_sa.html

Figure 265. Sales Assistant Builder — Select Template for Viewing Panel

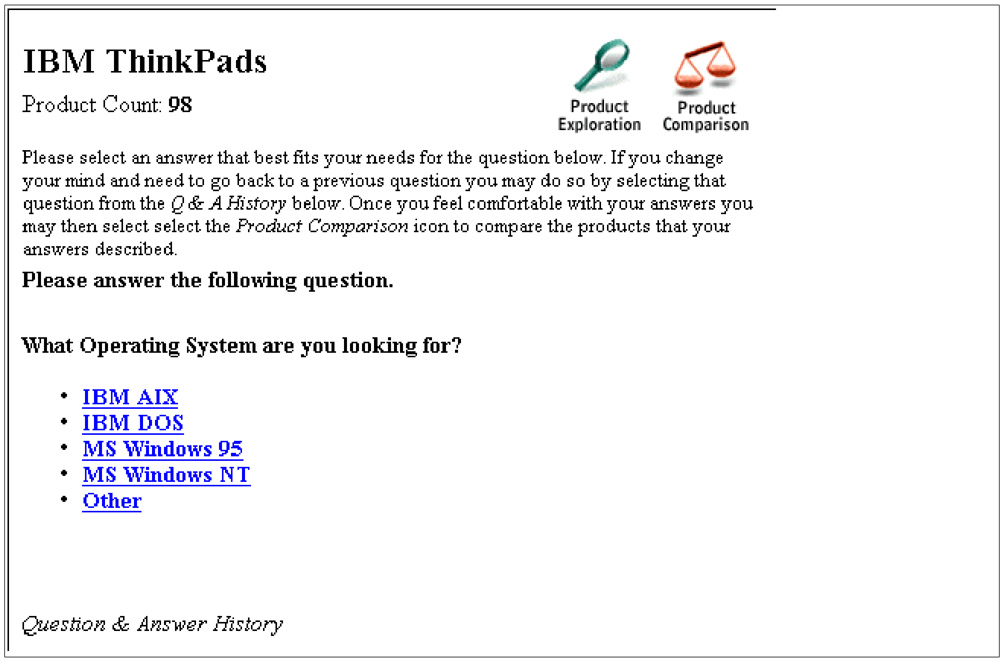
15.Select **File —> Save and View in Browser** (or click the glasses toolbar button) as shown in Figure 266, to save and view the Sales Assistant page in a browser.


Sales Assistance Builder - ShopITSO
File
Edit
Link
View
Help
Save
Import...
Save and View in Browser
Select Template for Viewing...
Exit
Product Count: 98
Unsigned Java Applet Window

	Order	Products
...u looking for?		
	1	7
IBM ThinkPads - (Product Exploration)		
IBM DOS	2	32
IBM ThinkPads - (Product Exploration)		
MS Windows 95	3	11
IBM ThinkPads - (Product Exploration)		
MS Windows NT	4	8
IBM ThinkPads - (Product Exploration)		
Other	5	40
IBM ThinkPads - (Product Exploration)		

Figure 266. Sales Assistant Builder — Viewing Sales Assistant Page in a Browser

16.The Sales Assistant page, shown in Figure 267 on page 298, pops up in a browser. Notice how the Sales Assistant page ask the questions to determine where to go next. Close the browser to continue working with Sales Assistant definition.



IBM ThinkPads

Product Count: 98

Please select an answer that best fits your needs for the question below. If you change your mind and need to go back to a previous question you may do so by selecting that question from the *Q & A History* below. Once you feel comfortable with your answers you may then select select the *Product Comparison* icon to compare the products that your answers described.

Please answer the following question.

What Operating System are you looking for?

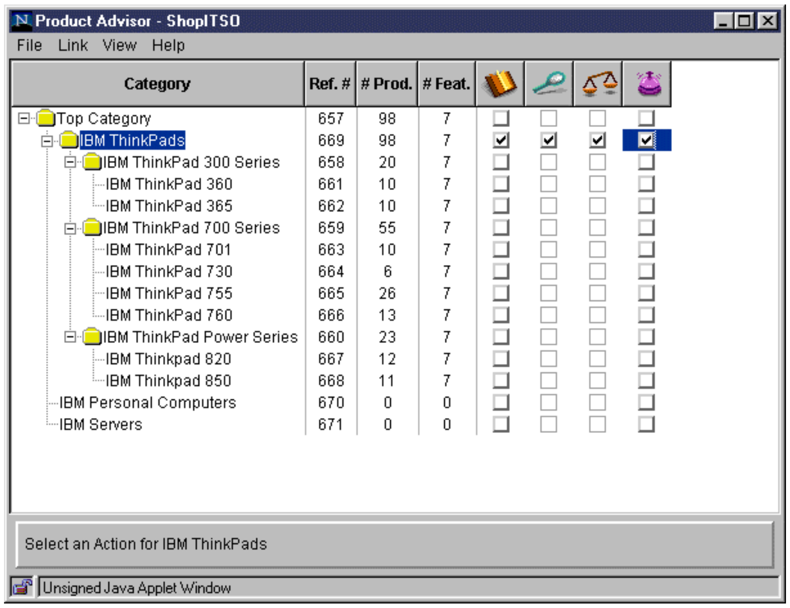
- IBM AIX
- IBM DOS
- MS Windows 95
- MS Windows NT
- Other

Question & Answer History

Figure 267. Sales Assistant Page

17. Select **File —> Exit** (or click the X in the top corner of the window) to exit the Sales Assistant Builder and return to the Product Advisor.

    Notice that in the Product Advisor window (Figure 268), which is now the status icon for the IBM ThinkPads category for the Product Exploration, the Product Comparison and Sales Assistant metaphors are changed from Enabled (empty box) to "Prepared" (clicked box).



Category	Ref. #	# Prod.	# Feat.				
Top Category	657	98	7	☐	☐	☐	☐
IBM ThinkPads	669	98	7	✓	✓	✓	✓
IBM ThinkPad 300 Series	658	20	7	☐	☐	☐	☐
IBM ThinkPad 360	661	10	7	☐	☐	☐	☐
IBM ThinkPad 365	662	10	7	☐	☐	☐	☐
IBM ThinkPad 700 Series	659	55	7	☐	☐	☐	☐
IBM ThinkPad 701	663	10	7	☐	☐	☐	☐
IBM ThinkPad 730	664	6	7	☐	☐	☐	☐
IBM ThinkPad 755	665	26	7	☐	☐	☐	☐
IBM ThinkPad 760	666	13	7	☐	☐	☐	☐
IBM ThinkPad Power Series	660	23	7	☐	☐	☐	☐
IBM Thinkpad 820	667	12	7	☐	☐	☐	☐
IBM Thinkpad 850	668	11	7	☐	☐	☐	☐
IBM Personal Computers	670	0	0	☐	☐	☐	☐
IBM Servers	671	0	0	☐	☐	☐	☐

Select an Action for IBM ThinkPads

Unsigned Java Applet Window

Figure 268. Product Advisor — Catalog and All Metaphors Prepared

18. Select **File—>Exit** (or click the X in the top corner of the window) to exit Product Advisor.

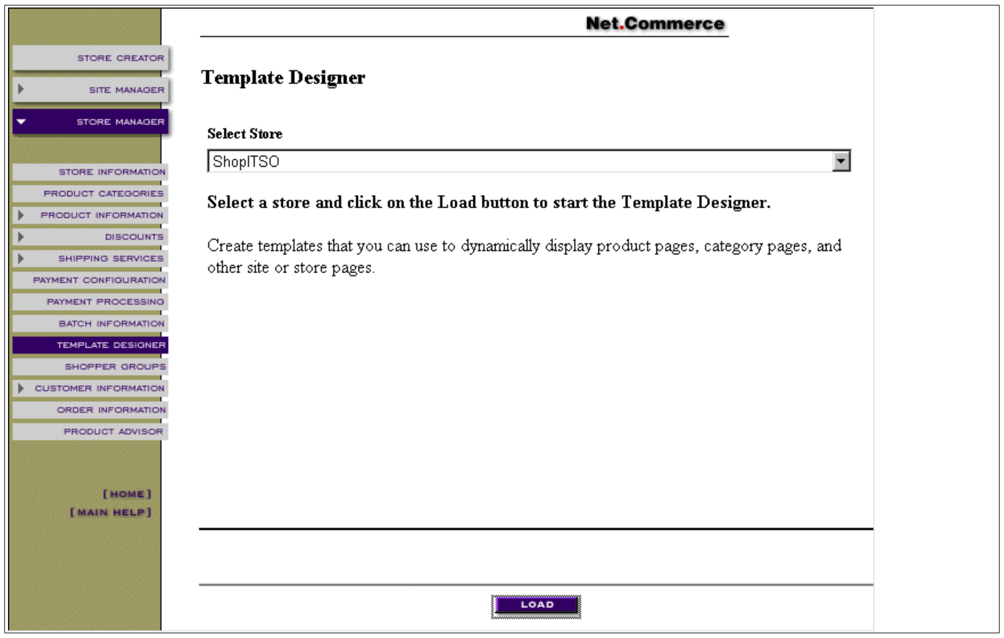
14.4 Using Template Designer to Customize Product Advisor Pages

After we set up Product Advisor, we create pages to link to the metaphors. Net.Commerce has sample templates included that can be used with Product Advisor. For you to use these templates with your implementation, you must customize them. We chose Template Designer as the tool to use for this task. Complete the following steps:

1. Login as `ncadmin` in the Net.Commmerce Administrator Web page at the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

2. From the Template Designer page in the Store Manager view, shown in Figure 269, select the **ShopITSO** store. Load the Template Designer applet by clicking the **Load** button. Two windows pop up: the Template Designer Status window and the Template Designer window.



Template Designer

Select Store

ShopITSO

Select a store and click on the Load button to start the Template Designer.

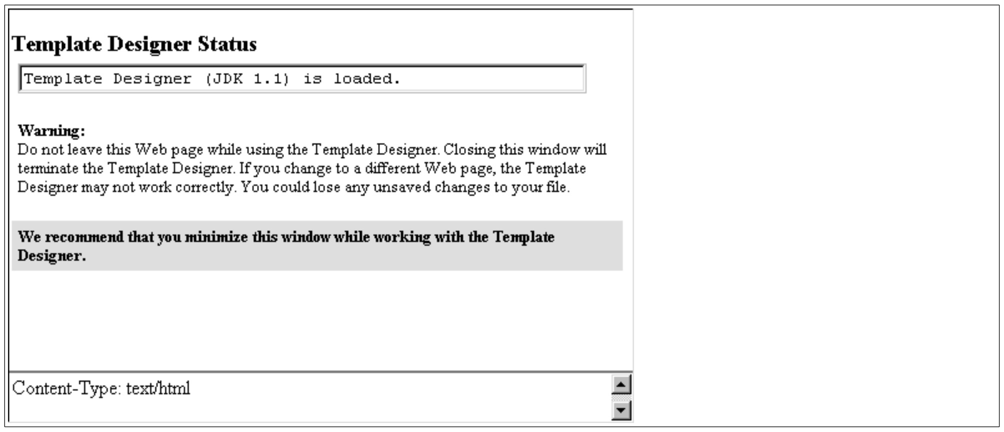
Create templates that you can use to dynamically display product pages, category pages, and other site or store pages.

Figure 269. Template Designer Page in the Net.Commerce Administrator

3. Minimize the Template Designer Status window, shown in Figure 270 on page 300, while working with the Template Designer window.

Important

Do not close the Template Designer Status window. If you close this window, the Template Designer window will also close!


Template Designer Status

Template Designer (JDK 1.1) is loaded.

Warning:
Do not leave this Web page while using the Template Designer. Closing this window will terminate the Template Designer. If you change to a different Web page, the Template Designer may not work correctly. You could lose any unsaved changes to your file.

We recommend that you minimize this window while working with the Template Designer.

Content-Type: text/html

Figure 270. Template Designer Status Window

14.4.1 Building the Base Pages for Product Advisor

The first step in customizing the Product Advisor pages is to create the three pages we will use for the metaphors. These are created from the samples shipped with Net.Commerce. You should repeat the following procedure three times to build the three pages for the metaphors.

Use Table 17 as a guide to substitute the names of the template (tmplsamp) and save as name (saveas) in the following procedure. The names shown in the table are the names we used in our sample store. You should change the names to reflect your store configuration.

Table 17. Names Used for SHOPITSO Templates

	Template Type	**Sample Name (tmplsamp)**	**Save as Name (saveas)**
_1.	Product Exploration	pe_te.html	shopitso_pe.html
_2.	Product Comparison	pc1_te.html	shopitso_pc.html
_3.	Sales Assistant	sa_te.html	shopitso_sa.html

Complete the following steps:

1. From the Template Designer window, shown in Figure 271 on page 301, select **File —> Open...** (or click the folder toolbar button) to open a template.

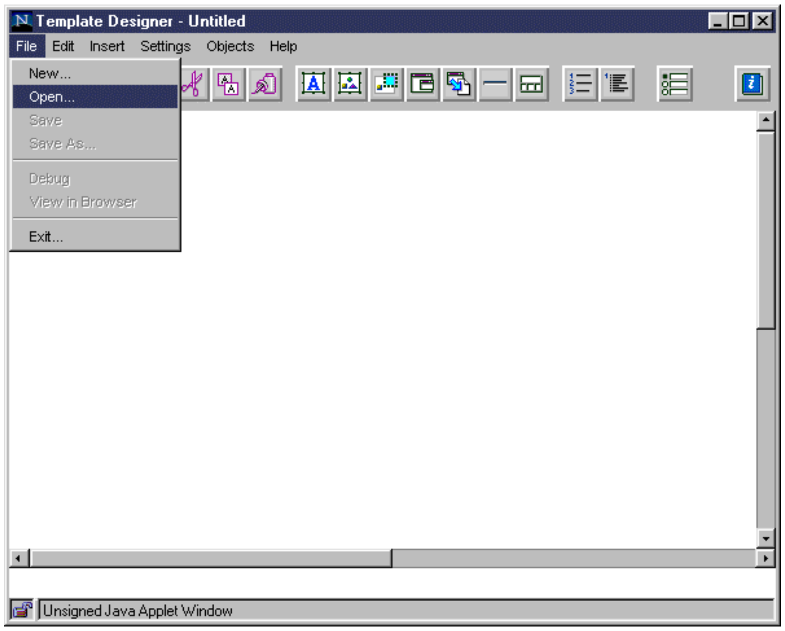

Template Designer - Untitled
File Edit Insert Settings Objects Help
New...
Open...
Save
Save As...
Debug
View in Browser
Exit...
Unsigned Java Applet Window

Figure 271. Template Designer Window — Open File

2. From the open file panel, shown in Figure 272, select the product advisor file type. Then, select the file name (**tmplsamp** from Table 17 on page 300). Then, click the **OK** button to open the sample Product Exploration template.

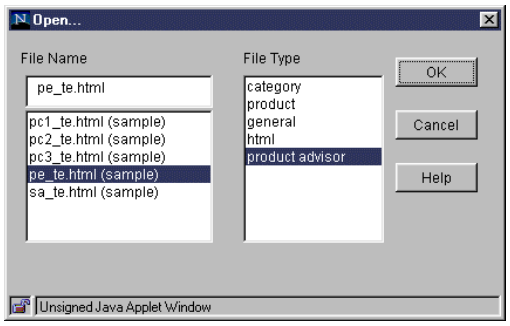
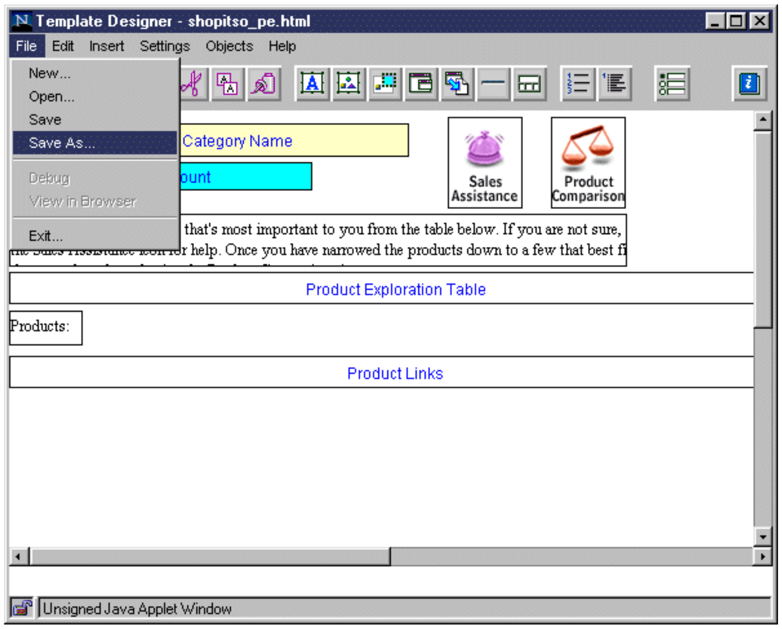


Figure 272. Template Designer — Open Product Exploration Sample Template

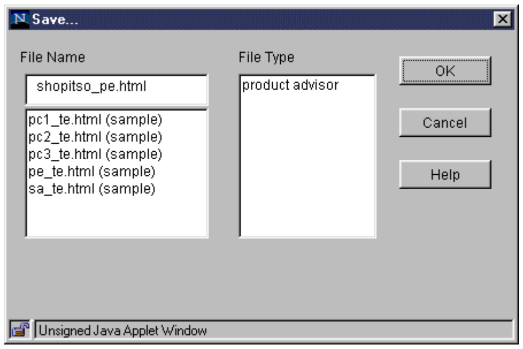
3. Select **File —> Save As...** to save the template with a different name, as shown in Figure 273 on page 302.



that's most important to you from the table below. If you are not sure, r help. Once you have narrowed the products down to a few that best fi

Figure 273. Product Exploration — Saving the New Product Exploration Template

4. Enter the file name (`saveas` from Table 17 on page 300). Click **OK**, as shown in Figure 274.



shopitso_pe.html

pc1_te.html (sample)
pc2_te.html (sample)
pc3_te.html (sample)
pe_te.html (sample)
sa_te.html (sample)

product advisor

Figure 274. Product Exploration Template — Save Panel

5. Repeat steps one through four until you build all of the required pages.

After you create the base pages, customize them as described in the following section.

14.4.2 Customizing the Product Exploration Page

This section describes how to customize the Product Exploration page. Perform the following series of steps to make the required changes:

1. From the Template Designer window, shown in Figure 275 on page 303, select **File —> Open...** (or click the folder toolbar button) to open a template.

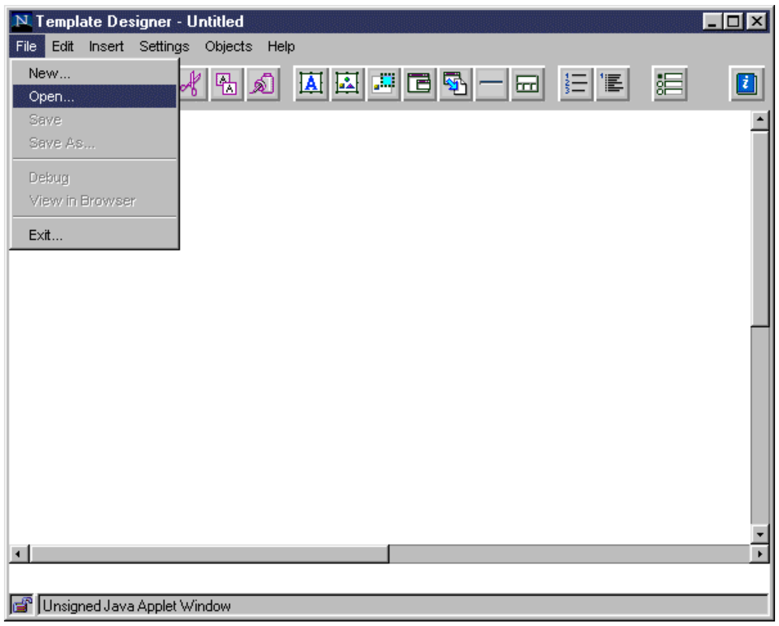
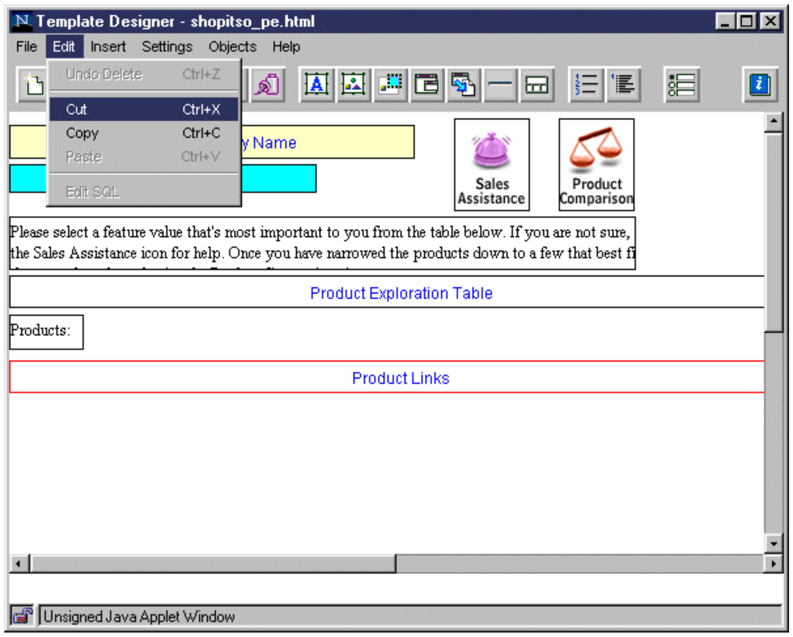


Figure 275. Template Designer Window — Open File

2. From the open file panel, select the **product advisor** file type. Then, select the **shopitso_pe.html** file name. Click the **OK** button to open the Product Exploration template.

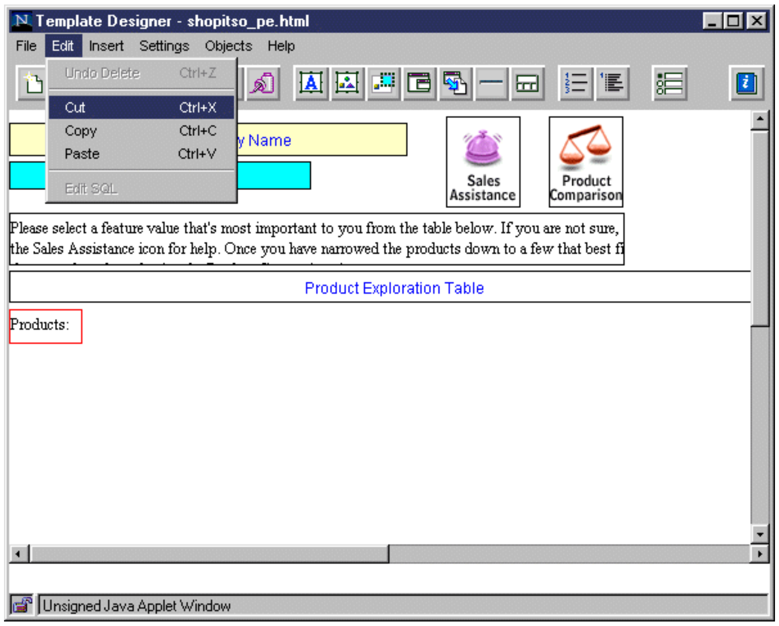
3. Click on the product links object labeled **Product Links** to select it, as shown in Figure 276. Then, select **Edit —> Cut** (or click the scissors toolbar button) to remove the object from the template.



Please select a feature value that's most important to you from the table below. If you are not sure, the Sales Assistance icon for help. Once you have narrowed the products down to a few that best fi

Figure 276. Product Exploration Template — Product Links Object

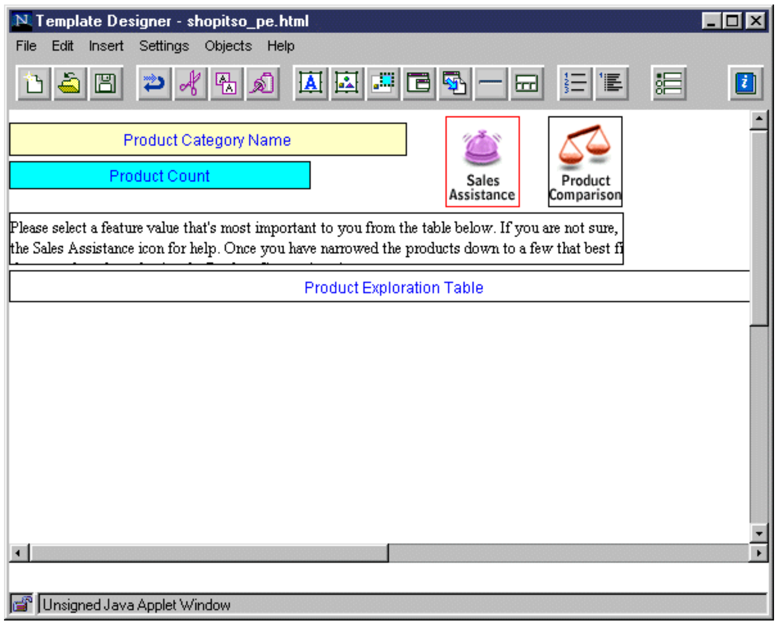
4. Click on the text object that contains the text **Products:** to select it (see Figure 277). Then, select **Edit —> Cut** (or click the scissors toolbar button) to remove the object from the template.



Please select a feature value that's most important to you from the table below. If you are not sure, the Sales Assistance icon for help. Once you have narrowed the products down to a few that best fi

Figure 277. Product Exploration Template — Text Object

5. Double-click the image object that contains the Sales Assistant image, as shown in Figure 278, to open the object's panel.



Please select a feature value that's most important to you from the table below. If you are not sure, the Sales Assistance icon for help. Once you have narrowed the products down to a few that best fi

Figure 278. Product Exploration Template — Sales Assistant Image Object

6. From the image object panel, shown in Figure 279, click the **Special Link...** button to change the target link of the object.

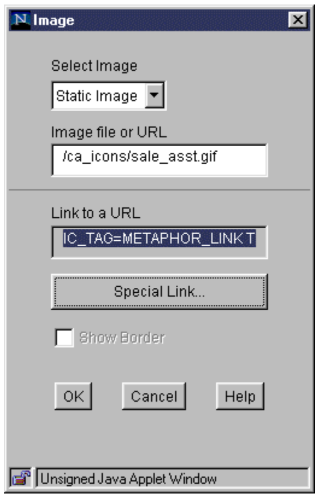


Figure 279. Product Exploration Template — Image Object Panel

7. From the object link panel, shown in Figure 280, select **Product Advisor Page** for the type of link. Then, select **/ca_html/shopitso_sa.html** for the template file to link to. Click the **OK** button to save the changes and return to the image object panel.

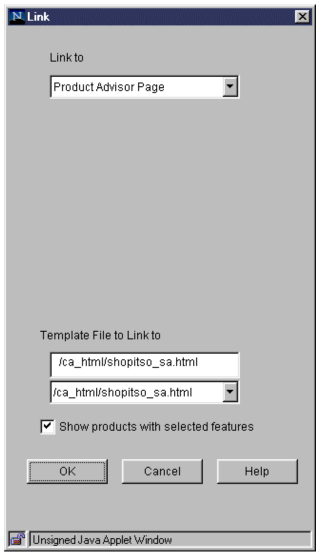

Link
Link to
Product Advisor Page
Template File to Link to
/ca_html/shopitso_sa.html
/ca_html/shopitso_sa.html
Show products with selected features
OK
Cancel
Help
Unsigned Java Applet Window

Figure 280. Product Exploration Template — Object Link Panel

8. From the image object panel, shown in Figure 281 on page 306, click the **OK** button to save the changes and return to the Template Designer.

   Notice that the Link to a URL field now points to the link just defined. To see the complete URL, place the cursor in the Link to URL field and use the arrow keys.

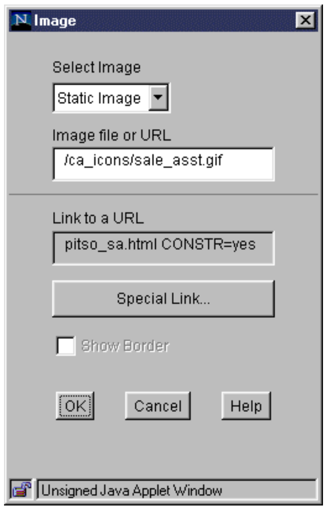


Figure 281. Product Exploration Template — Image Object Panel

9. Repeat steps five through eight for the Product Comparison image object. Use /ca_html/shopitso_pc.html for the link file in the object link panel, as shown in Figure 282.

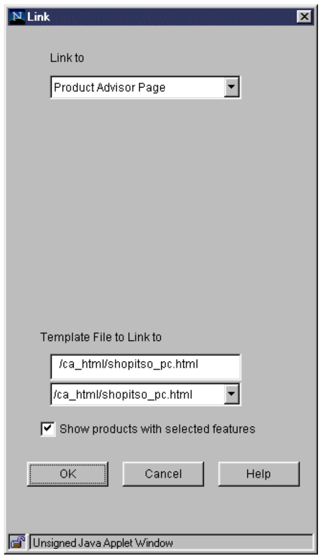
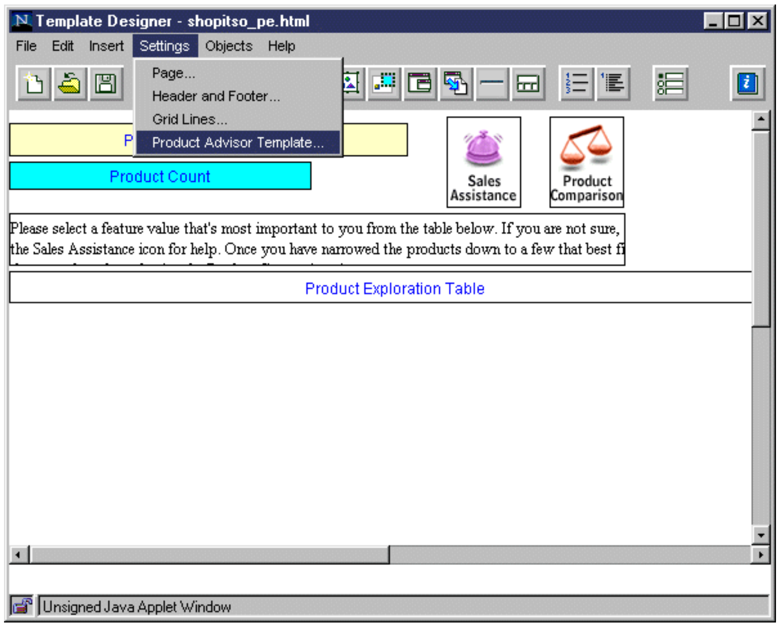

Link
Link to
Product Advisor Page
Template File to Link to
/ca_html/shopitso_pc.html
/ca_html/shopitso_pc.html
Show products with selected features
OK
Cancel
Help
Unsigned Java Applet Window

Figure 282. Product Exploration Template — Object Link Panel

10. Select **Settings —> Product Advisor Template...**, as shown in Figure 283 on page 307, to open the template panel.


Template Designer - shopitso_pe.html
File Edit Insert Settings Objects Help
Page...
Header and Footer...
Grid Lines...
Product Advisor Template...
Product Count
Sales Assistance
Product Comparison
Product Exploration Table
Unsigned Java Applet Window

Please select a feature value that's most important to you from the table below. If you are not sure, the Sales Assistance icon for help. Once you have narrowed the products down to a few that best fi

Figure 283. Product Exploration Template — Opening the Product Advisor Template

11. From the template panel, shown in Figure 284, select **Left** for template alignment. Click **OK**.

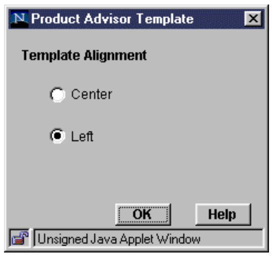


Figure 284. Product Exploration Template — Product Advisor Template Panel

12. Select **File —> Save** (or click the diskette toolbar button) to save the changes.

14.4.3 Customizing the Product Comparison Page

This section describes how to customize the Product Comparison page. Perform the following steps to make the required changes:

1. From the Template Designer window, shown in Figure 285 on page 308, select **File —> Open...** (or click the folder toolbar button) to open a template.

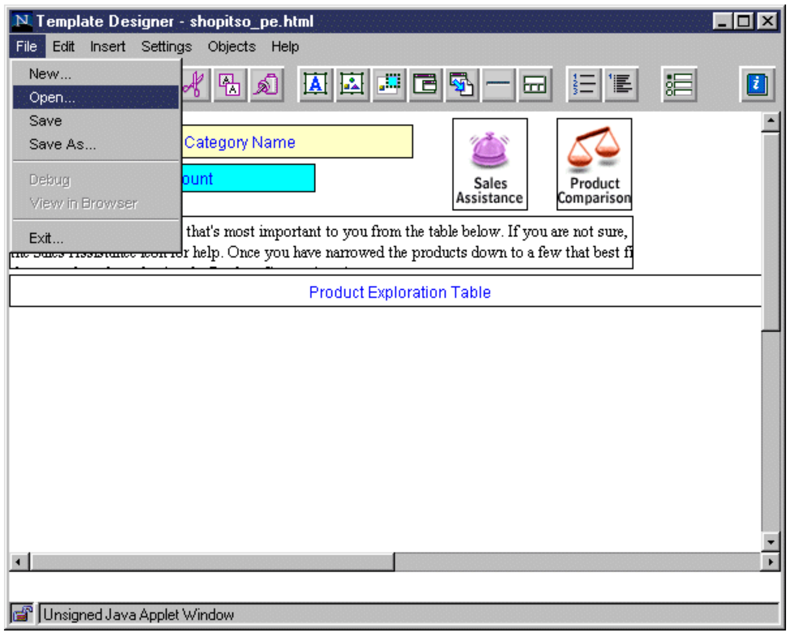

Template Designer - shopitso_pe.html
File Edit Insert Settings Objects Help
New...
Open...
Save
Save As...
Debug
View in Browser
Exit...
Category Name
ount
Sales Assistance
Product Comparison
that's most important to you from the table below. If you are not sure,
r help. Once you have narrowed the products down to a few that best fi
Product Exploration Table
Unsigned Java Applet Window

Figure 285. Template Designer Window — Open File

2. From the open file panel, select the product advisor file type. Then, select the **shopitso_pc.html** file name. Click the **OK** button to open your Product Comparison template.

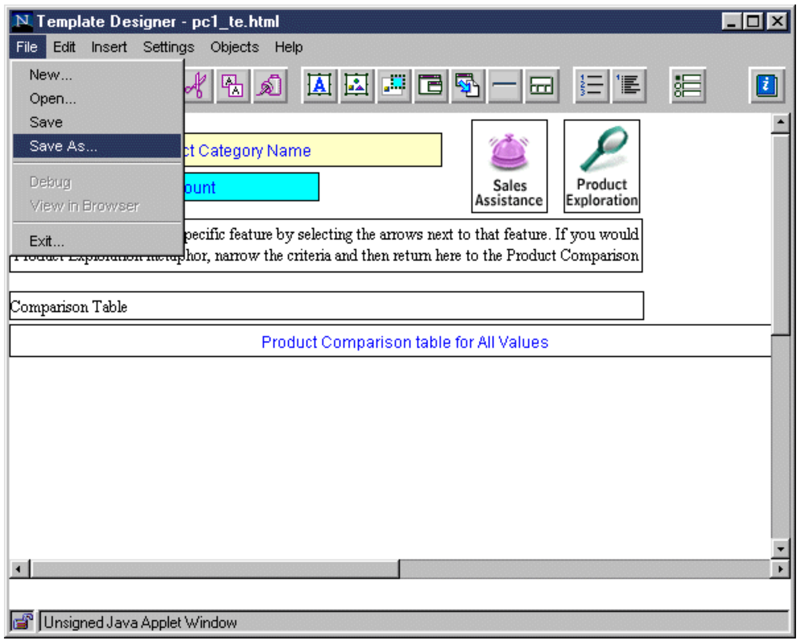
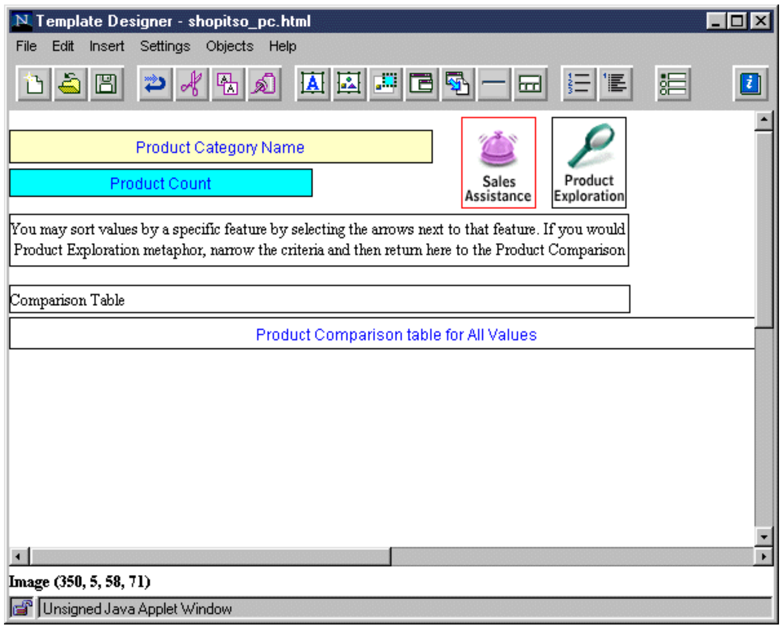

Template Designer - pc1_te.html
File Edit Insert Settings Objects Help
New...
Open...
Save
Save As...
Debug
View in Browser
Exit...
ct Category Name
ount
Sales Assistance
Product Exploration
pecific feature by selecting the arrows next to that feature. If you would
hor, narrow the criteria and then return here to the Product Comparison
Comparison Table
Product Comparison table for All Values
Unsigned Java Applet Window

Figure 286. Product Comparison — Saving New Product Comparison Template

3. Double-click the image object that contains the Sales Assistant image, as shown in Figure 287 on page 309, to open the object's panel.


Template Designer - shopitso_pc.html
File Edit Insert Settings Objects Help
Product Category Name
Product Count
Sales Assistance
Product Exploration

You may sort values by a specific feature by selecting the arrows next to that feature. If you would
Product Exploration metaphor, narrow the criteria and then return here to the Product Comparison

Comparison Table

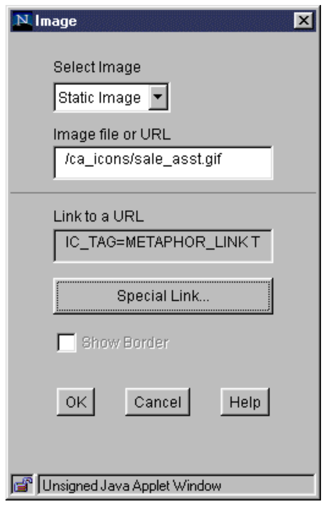
Product Comparison table for All Values

Image (350, 5, 58, 71)

Unsigned Java Applet Window

Figure 287. Product Comparison Template — Sales Assistant Image Object

4. From the image object panel, shown in Figure 288, click the **Special Link...** button to change the target link of the object.



Select Image: Static Image

Image file or URL: /ca_icons/sale_asst.gif

Link to a URL: IC_TAG=METAPHOR_LINKT

Show Border

Figure 288. Product Comparison Template — Image Object Panel

5. From the object link panel, shown in Figure 289 on page 310, select **Product Advisor Page** for the type of link. Then, select **/ca_html/shopitso_sa.html** for the template file to link. Click the **OK** button to save the changes and return to image object panel.

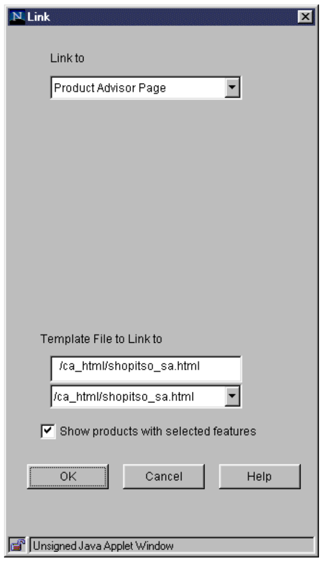

Link
Link to
Product Advisor Page
Template File to Link to
/ca_html/shopitso_sa.html
/ca_html/shopitso_sa.html
☑ Show products with selected features
OK
Cancel
Help
Unsigned Java Applet Window

Figure 289. Product Comparison Template — Object Link Panel

6. From the image object panel, as shown in Figure 290, click the **OK** button to save the changes and return to the Template Designer.

   Notice that the Link to a URL field now points to the link just defined. To see the complete URL, place the cursor in the Link to URL field and use the arrow keys.

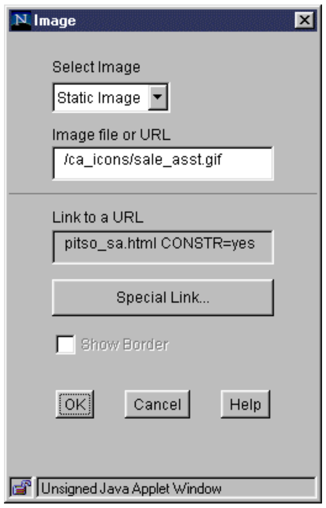
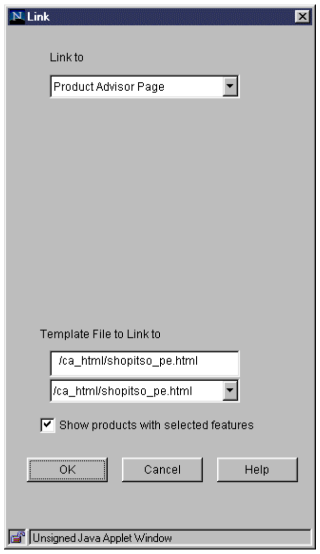


Figure 290. Product Comparison Template — Image Object Panel

7. Repeat steps three through six for the Product Exploration image object. Use /ca_html/shopitso_pe.html for the link file in the object link panel, as shown in Figure 291 on page 311.


Link

Link to

Product Advisor Page

Template File to Link to

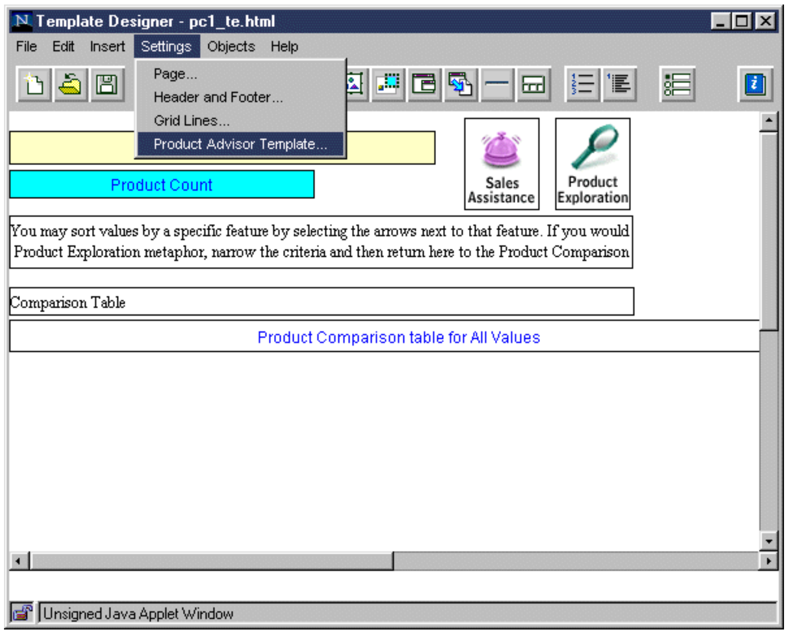
/ca_html/shopitso_pe.html

/ca_html/shopitso_pe.html

[✓] Show products with selected features

Figure 291. Product Comparison Template — Object Link Panel

8. Select **Settings —> Product Advisor Template...**, as shown in Figure 292, to open the template panel.


Template Designer - pc1_te.html

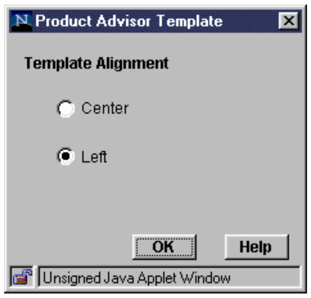
You may sort values by a specific feature by selecting the arrows next to that feature. If you would Product Exploration metaphor, narrow the criteria and then return here to the Product Comparison

Comparison Table

Product Comparison table for All Values

Figure 292. Product Comparison Template — Opening the Product Advisor Template

9. From the template panel, shown in Figure 293 on page 312, select **Left** for template alignment, and click **OK**.



Product Advisor Template

Template Alignment

( ) Center

(●) Left

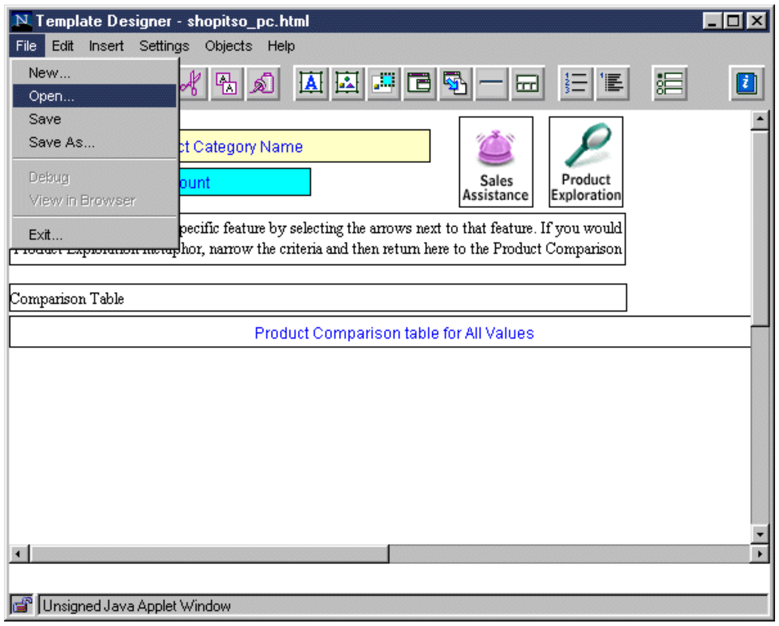
Figure 293. Product Comparison Template — Product Advisor Template Panel

10.Select **File —> Save** (or click the diskette toolbar button) to save the changes.

14.4.4 Customizing the Sales Assistant Pages

This section describes how to customize the Product Comparison page. Perform the following procedure to make the required changes:

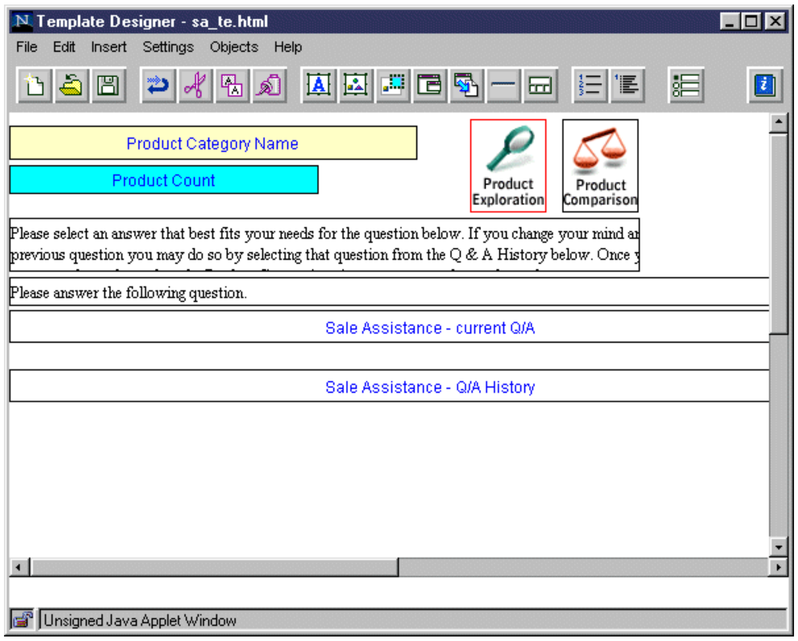
1. From the Template Designer window, shown in Figure 294, select **File —> Open...** (or click the folder toolbar button) to open a template.



pecific feature by selecting the arrows next to that feature. If you would ...hor, narrow the criteria and then return here to the Product Comparison

Figure 294. Template Designer — Open File

2. From the open file panel select the product advisor file type, the select the **shopitso_sa.html** file name. Click the **OK** button to open your Sales Assistant template.
3. Double-click on the image object that contains the Product Exploration image, shown in Figure 295 on page 313, to open the object's panel.

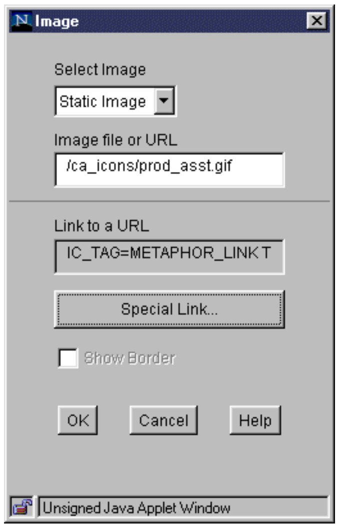

Template Designer - sa_te.html
File Edit Insert Settings Objects Help
Product Category Name
Product Count
Product Exploration
Product Comparison
Sale Assistance - current Q/A
Sale Assistance - Q/A History
Unsigned Java Applet Window

Please select an answer that best fits your needs for the question below. If you change your mind a previous question you may do so by selecting that question from the Q & A History below. Once y

Please answer the following question.

Figure 295. Sales Assistant Template — Product Exploration Image Object

4. From the image object panel, as shown in Figure 296, click the **Special Link...** button to change the target link of the object.



Select Image: Static Image

Image file or URL: /ca_icons/prod_asst.gif

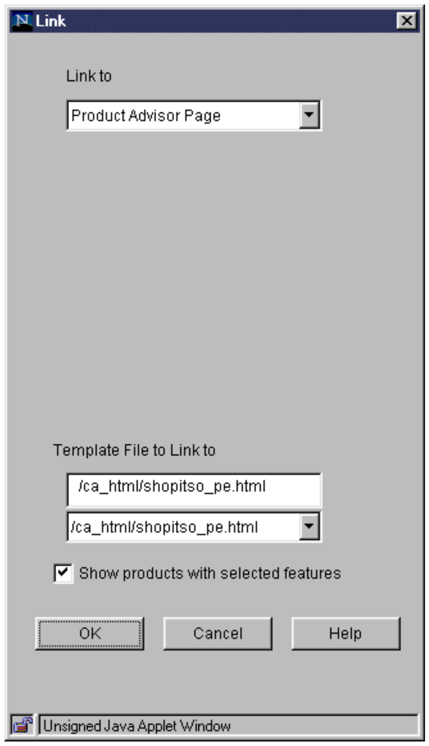
Link to a URL: IC_TAG=METAPHOR_LINKT

Special Link...

[ ] Show Border

Figure 296. Sales Assistant Template — Image Object Panel

5. From the object link panel, shown in Figure 297 on page 314, select **Product Advisor Page** for the type of link. Select **/ca_html/shopitso_pe.html** for the template file to which to link. Click the **OK** button to save the changes and return to the image object panel.


Link

Link to: Product Advisor Page

Template File to Link to: /ca_html/shopitso_pe.html

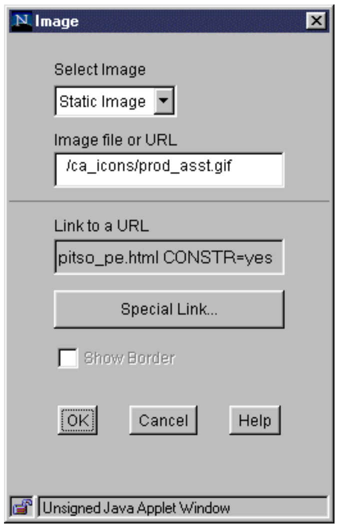
/ca_html/shopitso_pe.html

[✓] Show products with selected features

Figure 297. Sales Assistant Template — Object Link Panel

6. From the image object panel, shown in Figure 298, click the **OK** button to save the changes and return to the Template Designer.

   Notice that the Link to a URL field now points to the link just defined. To see the complete URL, place the cursor in the Link to URL field and use the arrow keys.



Image

Select Image: Static Image

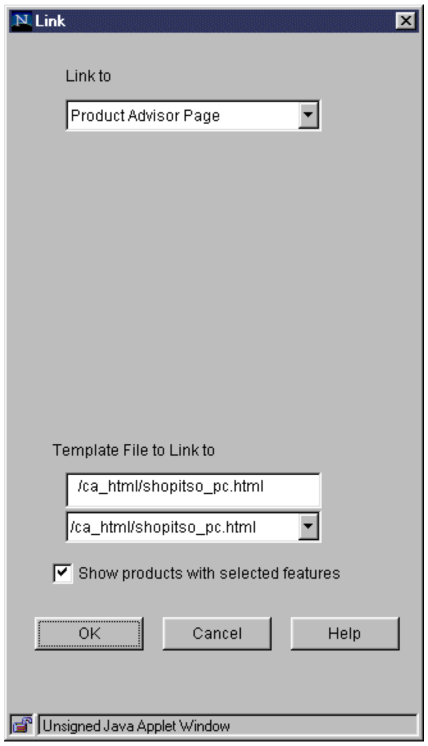
Image file or URL: /ca_icons/prod_asst.gif

Link to a URL: pitso_pe.html CONSTR=yes

[ ] Show Border

Figure 298. Sales Assistant Template — Image Object Panel

7. Repeat steps three through six for the Product Comparison image object. Use /ca_html/shopitso_pc.html for the link file in the object link panel, as shown in Figure 299 on page 315.


Link

Link to: Product Advisor Page

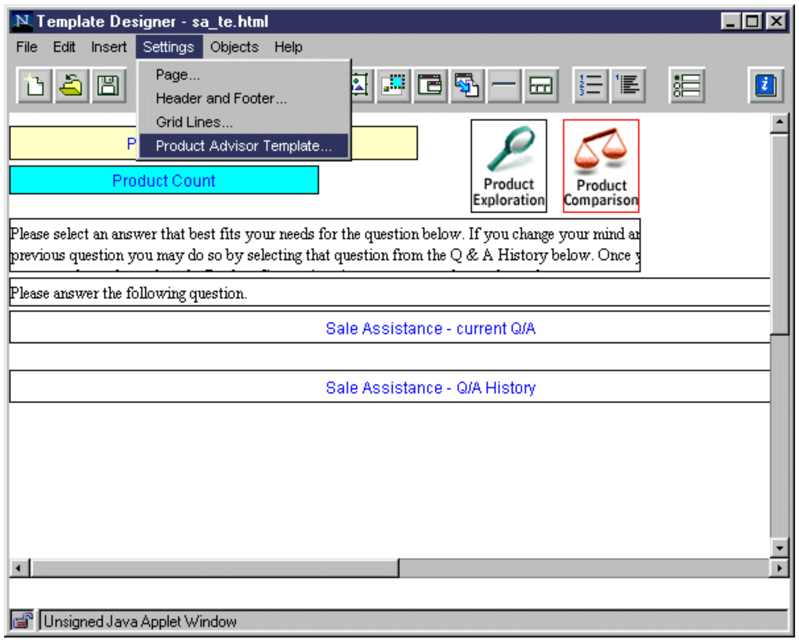
Template File to Link to: /ca_html/shopitso_pc.html

/ca_html/shopitso_pc.html

[✓] Show products with selected features

Figure 299. Sales Assistant Template — Object Link Panel

8. Select **Settings —> Product Advisor Template...**, as shown in Figure 300, to open the template panel.


Template Designer - sa_te.html

Please select an answer that best fits your needs for the question below. If you change your mind a
previous question you may do so by selecting that question from the Q & A History below. Once y

Please answer the following question.

Sale Assistance - current Q/A
Sale Assistance - Q/A History
Unsigned Java Applet Window

Figure 300. Sales Assistant Template — Opening the Product Advisor Template Panel

9. From the template panel, as shown in Figure 301 on page 316, select **Left** for the template alignment, and click **OK**.

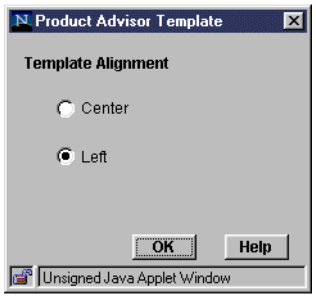


Figure 301. Sales Assistant Template — Product Advisor Template Panel

10.Select **File —> Save** (or click the diskette toolbar button) to save the changes.

11.Select **File —> Exit** (or click the X in the upper right corner of the window) as shown in Figure 302, to exit the Template Designer.

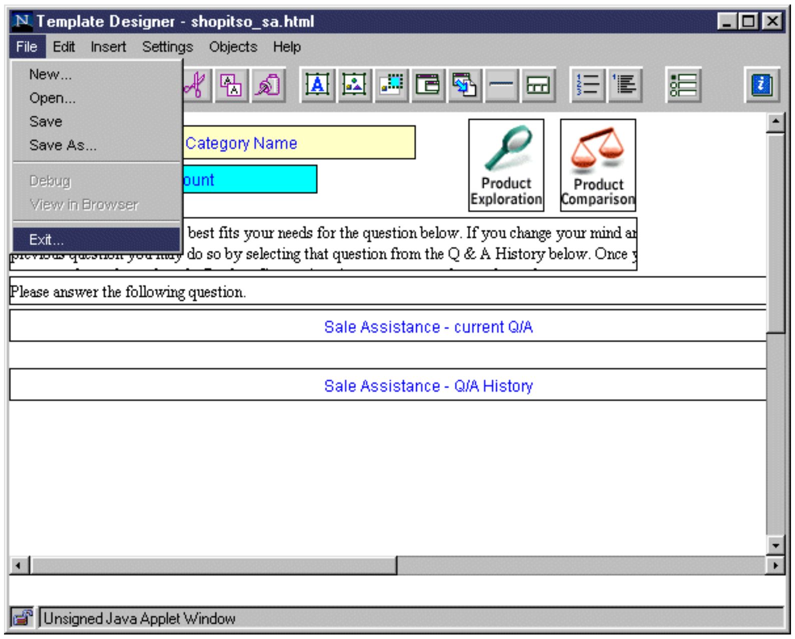

Template Designer - shopitso_sa.html
File Edit Insert Settings Objects Help
New...
Open...
Save
Save As...
Debug
View in Browser
Exit...
Category Name
Product Exploration
Product Comparison
Please answer the following question.
Sale Assistance - current Q/A
Sale Assistance - Q/A History
Unsigned Java Applet Window

Figure 302. Sales Assistant Template — Exiting Template Designer

12.The question is asked: `Are you are sure you want to exit Template Designer?` Click **Yes**.

14.5 Publishing Product Advisor Pages

To automate the process of publishing the Product Advisor metaphors defined for the IBM ThinkPads category, we used the following approach:

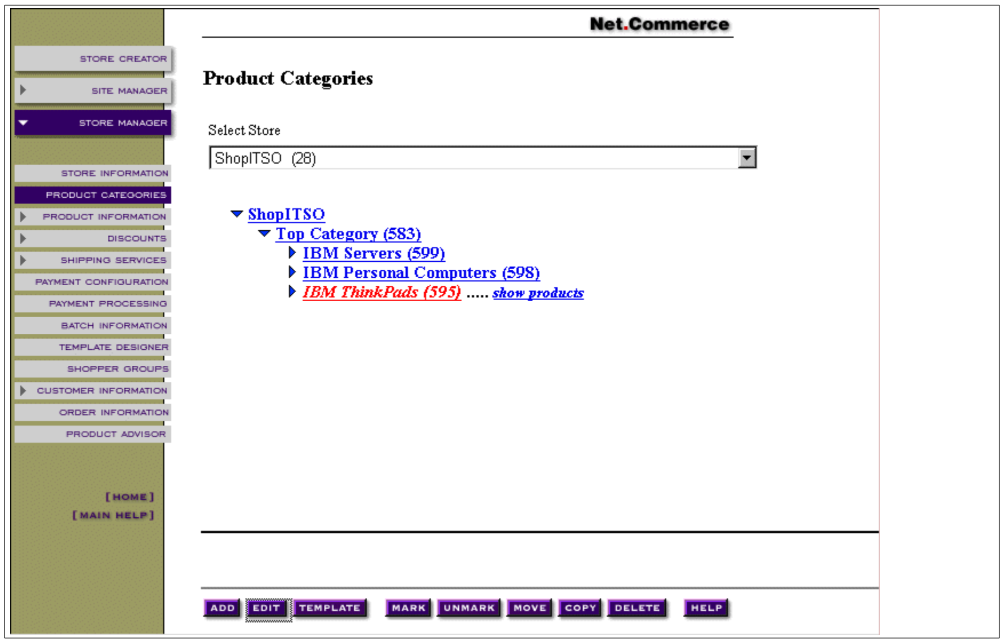
1. Use the category CustomField1 as a flag (resides in the Shopper Group Category Template table (CATESGP):
   - **Value of 1** — Product Advisor metaphors must be published
   - **Other value** — No Product Advisor metaphor to publish
2. Modify the cat0.d2w category macro to add a URL link to the Product Exploration metaphor when the category CustomField1 has the value of 1.

To implement this approach, follow these steps:

1. Login as `ncadmin` in the Net.Commerce Administration Web page at the following URL:

   `http://`*`fullyqualifiedhost_name`*`/ncadmin/`

2. From the Product Categories page in the Store Manager view, as shown in Figure 303, select the **ShopITSO** store. Click on **IBM ThinkPads** to select it. Then, click the **Edit** button.



Product Categories

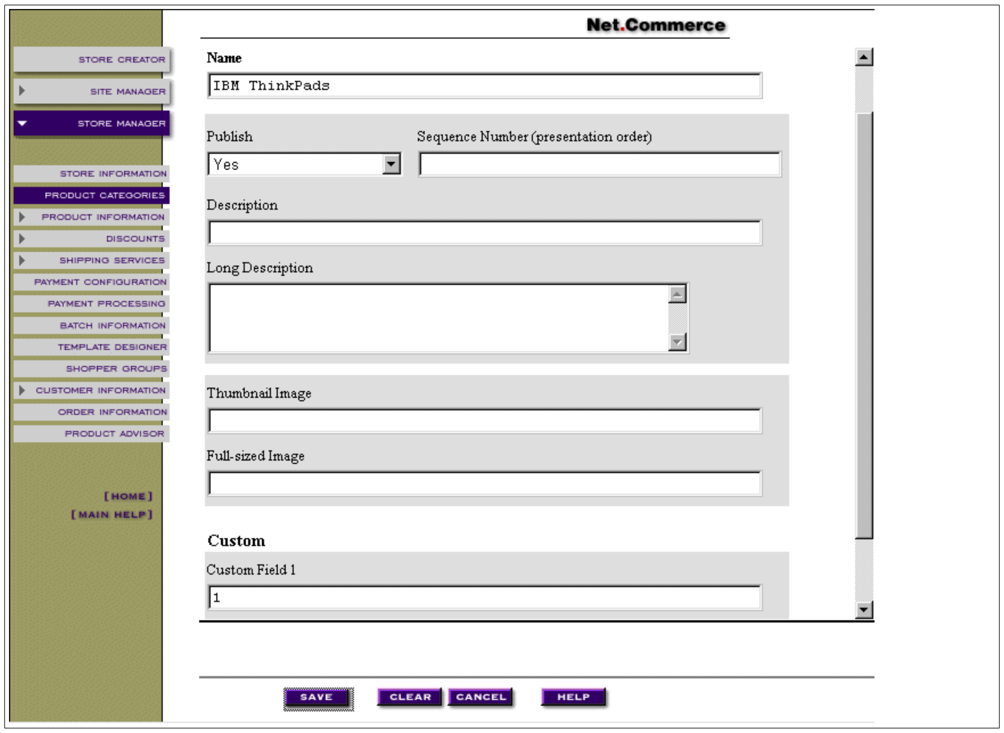
Select Store

ShopITSO (28)

- ShopITSO
  - Top Category (583)
    - IBM Servers (599)
    - IBM Personal Computers (598)
    - IBM ThinkPads (595) ..... show products

Figure 303. Product Categories — Editing Category with Product Advisor Metaphors

3. Set Custom Field 1 value to **1**, as shown in Figure 304 on page 318. Click **Save** to save the changes.



Name: IBM ThinkPads

Publish: Yes

Sequence Number (presentation order):

Description:

Long Description:

Thumbnail Image:

Full-sized Image:

Custom

Custom Field 1: 1

Figure 304. Product Categories — Setting Custom Field 1 to the Value of 1

4. Modify the ShopITSO/cat0.d2w category macro. Add the following code in the DISPLAY_CATEGORIES function:

```
%function(dtw_odbc) DISPLAY_CATEGORIES(){
    select CATEGORY.CGRFNBR, CATEGORY.CGMENBR, CATEGORY.CGNAME, CATEGORY.CGTHMB,
CATEGORY.CGFIELD1, 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{

<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR>
<TD ALIGN="left" VALIGN="top"><B><UL>
%ROW{
<LI><A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_CGRFNBR)&cgmenbr=$(V_CGMENBR)&CGRY_
NUM=$(CGRYNUM)"><FONT SIZE=2>$(V_CGNAME)</FONT></A>
%IF (V_CGFIELD1 == "1")
<A
HREF="/servlet/icviewer/ca_html/shopitso_pe.html?cgrfnbr=$(V_CGRFNBR)&cgmenbr=$(Me
rchantRefNum)" TARGET="main"><IMG SRC="/shopitso/padvisor.gif" ALT="Product
Exploration" BORDER="1" ALIGN="bottom"></A>
%ENDIF
</LI>

@DTW_ASSIGN(save_crpcgnbr, V_CRPCGNBR)
  %}
@DTW_assign(BACKUP, save_crpcgnbr)
</UL></B></TD>
</TR>
</TABLE>

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

Chapter 15. Importing Business Data into Net.Commerce

Many Net.Commerce customers already have their own back-end applications for running their business. These back-end applications access and use data in existing databases that contain information relating to customers, products, billing, and inventory control.

This chapter looks at the options available for loading data from our back-end database into our Net.Commerce database. It also discusses the important task of keeping the Net.Commerce data and the back-end system data fully synchronized.

When a Net.Commerce instance is created, the database created has sufficient information in it to allow the mall to be administrated. However, it has no product, category, or price information unless the option was taken to install a demo store or mall.

The Net.Commerce database tables contain all the information Net.Commerce needs to store information about the mall or store and its operation, such as products, categories, and prices. Initially these product, price, and category tables are empty, so this information must be fed into the Net.Commerce database. This information can be added to the database from the administrator screens. However, if there is a large amount of data to be added to the database, manual addition may be impractical. In this case, customers may want to load data from their current back-end system for the initial dataload phase and for ongoing changes.

15.1 General Considerations for Loading Data

The skills required and the method you use for loading the data into your store or mall depends on your data source and the skills you have available. If you are creating your store or mall for the first time, you can use the Net.Commerce tools, Store Creator and Administrator, to enter all of your store information from scratch. For most customers, this is not the case and you need to perform some form of data loading and synchronizing of the data from existing data sources.

15.1.1 Loading the Net.Commerce Database

When setting up an individual store in a Net.Commerce site, the store manager has the ability to enter a wide variety of data, including productspecific data. This can be done through the Net.Commerce administrator. Entering data for a single product requires filling in several "screens" worth of fields. This certainly works, but for a store that is to have thousands or even tens of thousands (or more) of items, this process is less than desirable. Fortunately, Net.Commerce includes a utility to incorporate existing data without having to manually key the information through the Net.Commerce store administrator panels. This utility, called *Mass Import*, is discussed in 15.2.2, “Mass Import” on page 321. In addition to using Net.Commerce's Mass Import utility, other techniques for populating the Net.Commerce files are also available. These techniques include using platform-specific utilities (such as DFU (data file utility)) on the AS/400 system, generic DB/2 tools (such as DataPropagator Relational x.x for AS/400), tools such as NotesPump, or writing a custom program to accomplish the task. Finally, simply populating the Net.Commerce files one time may not be all that is required to integrate Net.Commerce and legacy applications. A customer may continue to use the legacy applications in a non-Web environment, and require the Net.Commerce data to be synchronized with the non-Net.Commerce data.

15.2 Options for Loading Data

There are two possible ways to load data into the Net.Commerce database from the back-end system:

- Write your own program to write data directly into the Net.Commerce database.
- Use the Mass Import utility which is part of the Net.Commerce product.

Generally, you should try to avoid using any other method other than mass import‘. The only reason for using other methods is the performance penalty that you pay for using mass import. Some programming may be required to generate the input files to use with the Mass Import utility.

15.2.1 Writing Your Own Program to Import Data

If you do not want to use the mass import utility, then you can use one of the tools provided with the AS/400 system to populate the database. The tables related to each instance are stored in a library of the same name as the instance.

The ways of accessing a collection of tables on the AS/400 system are:

- Native database manipulation commands, such as Open Physical File, from an AS/400 supported language such as RPG
- Embedded SQL from a language supported on the AS/400 system
- An ODBC-compliant application tool, such as Powerbuilder or Delphi, in combination with the Client Access ODBC driver
- Interactive SQL using the STRSQL command
- A data replication product such as Data Propagator

However, if you are not using mass import, you must take care to ensure that referential integrity is satisfied. This can be quite complex and requires a deep understanding of the Net.Commerce data model. Furthermore, you have to handle any future changes in the Net.Commerce data model.

Using the direct approach can be useful when updating particular columns since it will run faster than the Mass Import utility. Plus, it will not require the additional step of creating the mass import input file.

Examples of updates that can benefit from this approach are:

- Updating price information for all the products. You can write a utility program or SQL query that copies the price values from a table in your legacy system into the PPPRC column of the PRODPRCS table.
- Updating merchant customizable fields.
- Updating product descriptions.

15.2.2 Mass Import

Mass import is a utility supplied with Net.Commerce that allows the Net.Commerce database to be populated from pre-formatted data in a text file. This utility is used to populate the demo mall Net.Commerce database if the option is chosen to install the demo mall when configuring a Net.Commerce instance.

The Mass Import utility allows you to populate the Net.Commerce database with information about categories and products. The utility imports data from a delimited flat file on the IFS (including /QSYS.LIB) into the product and category related tables in the DB2/400 database. The product and category related tables contain information about categories, products, items, and their inter-relationships. These tables are:

- CATEGORY — Category
- CATESGP — Shopper Group Category Template
- CGRYREL — Category Relationship
- PRODUCT — Product
- CGPRREL — Category Product Relationship
- PRODDSTATR — Product Distinct Attribute
- PRODATR — Product Attribute
- PRODPRCS — Product Price
- PRODSGP — Shopper Group Product Template

The Mass Import utility simplifies the process of importing data in the following ways:

- It automatically generates reference numbers to prevent referential integrity errors.
- It automatically checks whether or not records exist in the database. If a record exists, it updates the record; otherwise it inserts a new record.

The Mass Import utility is ideal for importing large quantities of product data into the database tables. If you continue to use your back-end system for entering product data, we also recommend that you use the mass import to synchronize the changes from the back-end tables to Net.Commerce tables.

If you have to insert or update individual rows directly into the Net.Commerce database, use the forms provided by the Net.Commerce Administrator function.

To import product and category data, there are two steps involved:

1. Create an import file. We provide an example of using custom ILE RPG program to extract data from our back-end system to an import file.
2. Run the Mass Import utility using the mass import file created in the previous step.

Please note that the Mass Import utility does not perform as well as directly interacting with the Net.Commerce database. However, it is the correct and *safest* way to download data from back-end systems to the Net.Commerce database.

15.2.2.1 Import File Creation

Before you can use the Mass Import utility, you must create an import file that contains the commands and the data with which to populate the product and category-related tables. The file is treated as a continuous string of transactions, with the columns and rows of data separated by a delimiter. The easiest way to create the import file is to write your own conversion script. We provide an example written in ILE RPG. You can also create the import file manually by using text editor such as Note Pad.

The following commands can be included in the import file:

#COLUMNDELIMITER	Defines the character that is used as a column delimiter.
#ROWDELIMITER	Defines the character that is used as a row delimiter.
#STORE	Specifies the name of the store for which the product and category records are being created.
#CATEGORY	Populates the Category table and the Category Relationship table.
#CATESGP	Populates the Shopper Group Category Template table.
#PRODUCT	Populates the Product table.
#CGPRREL	Populates the Category Product Relationship table.
#PRODDSTATR	Populates the Product Distinct Attribute table.
#PRODATR	Populates the Product Attribute table.
#PRODPRCS	Populates the Product Price table.
#PRODSGP	Populates the Shopper Group Product Template table.

All of the above commands can be specified more than once. For example, you can define the row delimiter as ",". Then, later in the file, you may need to add product transactions that contain this delimiter in the product descriptions. In this case, you can define a new row delimiter for all of the affected product transactions.

Note: Following each #STORE command, you must add the product and category commands to the import file in the sequence shown above. If you do not follow this sequence, referential integrity problems may result. You can type the commands in uppercase, lowercase, or mixed case, for example:

```
#PRODATR
#prodatr
#ProdAtr
```

If you want to leave a column in a transaction empty, you must still insert a delimiter to create a placeholder for that column. If you insert a new record into the database and you leave a column empty (if a default value exists as specified in the database schema), the Mass Import utility automatically inserts the default value into the column. If you update a record and you leave a column empty, the existing column value remains. Columns are updated only when you specify new values.

Figure 305 on page 323 shows an example of an import file.

```
#STORE;East West Food Mart
#CATEGORY;Water;;;;c1wa_n.gif;;1;c1wa_h.gif;;Beverages;1
#CATESGP;Water;;toc.d2w;;;;
#PRODUCT;100;;Evian;;;;wac1100a.gif;;1;;;;;;;FD01;;;;
#PRODDSTATR;100;Size
#PRODPRCS;100;;;USD
#PRODSGP;100;Frequent Shoppers;freqshp.d2w
#PRODSGP;100;Gold Club;goldclb.d2w;club for senior shoppers
#CGPRREL;Water;100;1
#PRODUCT;100-S;100;Evian;;;;wac1102a.gif;;1;;;;;;;FD01;;;999;
#PRODATR;100-S;Size;500 mL
#PRODPRCS;100-S;;1;USD
#PRODSGP;100-S;Frequent Shoppers;freqshp.d2w
#PRODSGP;100-S;Gold Club;goldclb.d2w;club for senior shoppers
#PRODUCT;100-M;100;Evian;;;;wac1102a.gif;;1;;;;;;;FD01;;;999;
#PRODATR;100-M;Size;1 L
#PRODPRCS;100-M;;2;USD
#PRODSGP;100-M;;;;
#PRODUCT;100-L;100;Evian;;;;wac1102a.gif;;1;;;;;;;FD01;;;999;
#PRODATR;100-L;Size;1.5 L
#PRODPRCS;100-L;;3;USD
#PRODSGP;100-L;Frequent Shoppers;freqshp.d2w
#PRODSGP;100-L;Gold Club;goldclb.d2w;club for senior shoppers
#PRODUCT;101;;San Pellegrino;;;;wac1101a.gif;;1;;;;;;;FD01;;;;
#PRODDSTATR;101;Size
#PRODPRCS;101;;;USD
#PRODSGP;101;Frequent Shoppers;freqshp.d2w
#PRODSGP;101;Gold Club;goldclb.d2w;club for senior shoppers
#CGPRREL;Water;101;2
#PRODUCT;101-S;101;San Pellegrino;;;;wac1101a.gif;;1;;;;;;;FD01;;;999;
#PRODATR;101-S;Size;500 mL
#PRODPRCS;101-S;;1;USD
#PRODSGP;101-S;Frequent Shoppers;freqshp.d2w
#PRODSGP;101-S;Gold Club;goldclb.d2w;club for senior shoppers
#PRODUCT;101-M;101;San Pellegrino;;;;wac1101a.gif;;1;;;;;;;FD01;;;999;
#PRODATR;101-M;Size;1 L
#PRODPRCS;101-M;;2;USD
#PRODSGP;101-Infrequent Shoppers;freqshp.d2w
#PRODSGP;101-M;Gold Club;goldclb.d2w;club for senior shoppers
#PRODUCT;101-L;101;San Pellegrino;;;;wac1101a.gif;;1;;;;;;;FD01;;;999;
#PRODATR;101-L;Size;1.5 L
#PRODPRCS;101-L;;3;USD
#PRODSGP;100-L;Frequent Shoppers;freqshp.d2w
#PRODSGP;100-L;Gold Club;goldclb.d2w;club for senior shoppers
#CATEGORY;Softdrinks;;;;c1so_n.gif;;1;c1so_h.gif;;Beverages;2
#CATESGP;Softdrinks;;toc.d2w;;;;
```

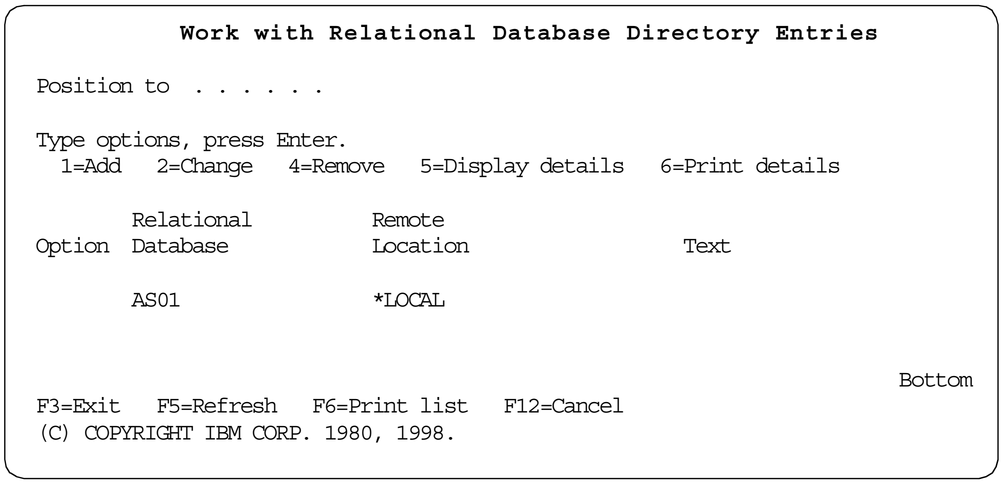
Figure 305. Sample Import File

Note: Do not import double quotes (") into the database.

One import file can contain many store directives, each populating a store in the required mall. A #STORE directive tells the Mass Import utility that the lines following it, until the next #STORE directive, are all for the store indicated. Mass import ensures that the correct referential integrity constraint information is generated so that more than one store can have the same category and product. A #STORE directive for a store can appear more than once in the file. Each store directive and the lines following it until the next #STORE directive can be separated into a separate input file for mass import. Thus, if you are creating these files from an existing database, you can generate one import file for all the categories in each store from one set of tables and then generate another input file for all the products in each store. However, you must make sure that no referential integrity constraints are violated by keeping the order of the import commands correct as discussed earlier. For example, you should not attempt to insert a new product before inserting the category it goes under.

15.2.2.2 Using the Mass Import Utility

Once you complete the import file, you can run the Mass Import utility. Before calling the Mass Import utility, you need to know the name of the local or remote database, as defined in the relational database directory of the AS/400 system. This will be the name of the database where the Net.Commerce tables are located. To work with database directory entries, type `WRKRDBDIRE`, and press **Enter**. A screen similar to the example in Figure 306 is displayed.

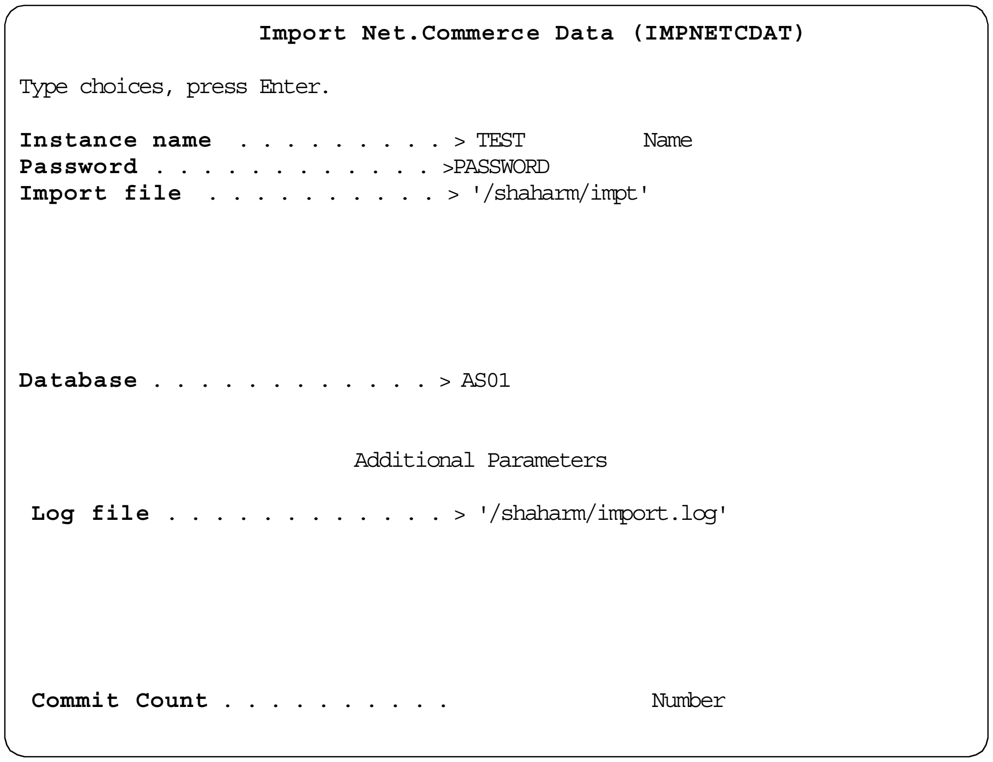


```
           Work with Relational Database Directory Entries

 Position to  . . . . . .

 Type options, press Enter.
   1=Add   2=Change   4=Remove   5=Display details   6=Print details

        Relational          Remote
 Option  Database            Location                 Text

        AS01                *LOCAL


                                                                Bottom
 F3=Exit   F5=Refresh   F6=Print list   F12=Cancel
 (C) COPYRIGHT IBM CORP. 1980, 1998.
```

Figure 306. WRKRDBDIRE — Checking the Location of a Database

If you are using the local database, then the relational database name corresponding to the location *LOCAL is the name of the database that you use. Otherwise, use the RDB name corresponding to the remote location you want. In our example, the local database is named AS01.

To run the Mass Import utility, use the Import Net.Commerce Data (`IMPNETCDAT`) command. The command is located in library QSYS. Type `IMPNETCDAT` and press **F4**. The command prompt, shown in Figure 307 on page 325, is displayed.



```
                 Import Net.Commerce Data (IMPNETCDAT)

Type choices, press Enter.

Instance name  . . . . . . . . . > TEST          Name
Password . . . . . . . . . . . . >PASSWORD
Import file  . . . . . . . . . . > '/shaharm/impt'







Database . . . . . . . . . . . . > AS01


                           Additional Parameters

 Log file . . . . . . . . . . . . > '/shaharm/import.log'






 Commit Count . . . . . . . . . .                 Number
```

Figure 307. IMPNETCDAT — Command Prompt

Table 18 describes the `IMPNETCDAT` command.

Table 18. Command IMPNETCDAT Parameters

Parameter name	Description
Instance name	The name of the Net.Commerce instance into which the data will be imported. This is the library and user profile name relating to the instance.
Password	The Net.Commerce instance database password. This is the instance user profile's password. This parameter is not required if the command is being invoked by the database owner. In our example, the user profile test will not have to type the password.
Import File	The name of the file that contains the product and category data to be imported. It can be a database file member or a stream file. Its path name must conform to the IFS directory naming conventions. For example, QSYS.LIB/LIBA.LIB/FILEA.FILE/ MBRA.MBR is the form required by the QSYS.LIB file system. If you use your own utility to create the import file, you may find it more convenient to store the import file in QSYS.LIB since you can view its contents more easily.

Parameter name	Description
Database	The name of the database into which the data is being imported. The Mass Import utility uses DRDA to connect to the database. This parameter refers to the database name found in the Relational Database Directory of the AS/400 system. In our example, we use the name AS01 since it is our *local rdb entry name.
Log File	The name of the file in which you want the Mass Import utility to record its activities. If this parameter is not specified, this default log file is created: /QIBM/Userdata/NetCommerce/instance/<instance_name>/logs/massimpt.log. The log file can be directed to QSYS.LIB. If the location of the log file is not on the /QSYS.LIB file system, you can view the log file contents by using the AS/400 command EDTF (available as a PTF).
Commit Count	The number of transactions within a commit scope. The default is 1. You can only enter integers for this parameter. If a commit count of 0 is specified, the data import is committed after all the records are processed and the database connection is terminated. You can improve performance by adjusting the commit count parameter. A higher commit count setting means that the database will be committed less frequently, which saves some time.

15.2.2.3 Checking the Results of the Mass Import Utility

Standard output is used by the Mass Import utility to report some general information. If you use the Mass Import utility interactively, you will see the results on your screen. If the mass import was submitted to batch, the results are directed to a spooled file. Figure 308 shows an example of the standard output produced as a result of running the IMPNETCDAT command.

```
Mass Import Utility for Net.Commerce Version 3.1
(c) Copyright IBM Corporation 1997, 1998. All rights reserved.
  19990312110615Fri Mar 12 11:06:15 1999
CMN1304I Started reading input file. Processing began.
  19990312110615CMN1601I For utility processing details, see the log file /sha
  19990312110615
Number of Successful Transactions : 55
Number of Failed Transactions : 0
  19990312110622
Fri Mar 12 11:06:22 1999
CMN1305I Finished reading input file. Processing completed.
  19990312110623
```

Figure 308. Mass Import Sample Log File

The mass import log file as entered in the IMPNETCDAT log file parameter contains detailed information with the description of any errors. Figure 309 shows an example of the mass import log file.

```
****************** Beginning of data ********************
   Fri Mar 12 11:21:32 1999 CMN1304I Started reading input file. Processing be
     19990312112132CMN1303I Connected successfully to the database.
     19990312112132CMN1310I Number of Transactions processed : 1
     19990312112132CMN1310I Number of Transactions processed : 2
     19990312112132CMN1310I Number of Transactions processed : 3
     19990312112132CMN1043E Product Price table update failed, (SQL class) ret
   UPDATE prodprcs set ppsgnbr=NULL,ppprc=ZZZ0.00,ppcur='USD',pppre=0,ppdeffs=
      19990312112133CMN1302I The following transaction failed...
   [00004]#PRODPRCS;1;;ZZZ0.00;USD;;;;&
     19990312112133
   CMN1312W Any uncommitted transactions within the commit scope are rolled ba
     19990312112133

   CMN1035E Product '88' does not exist in the table PRODUCT.
    SELECT PRRFNBR FROM PRODUCT WHERE PRNBR='88' AND PRMENBR=2067

   ******
   STATUS CMN0003S: Database 'AS01' has been commited.
   ******

     19990312112136

   Number of Successful Transactions : 47
   Number of Failed Transactions : 3



     19990312112136
```

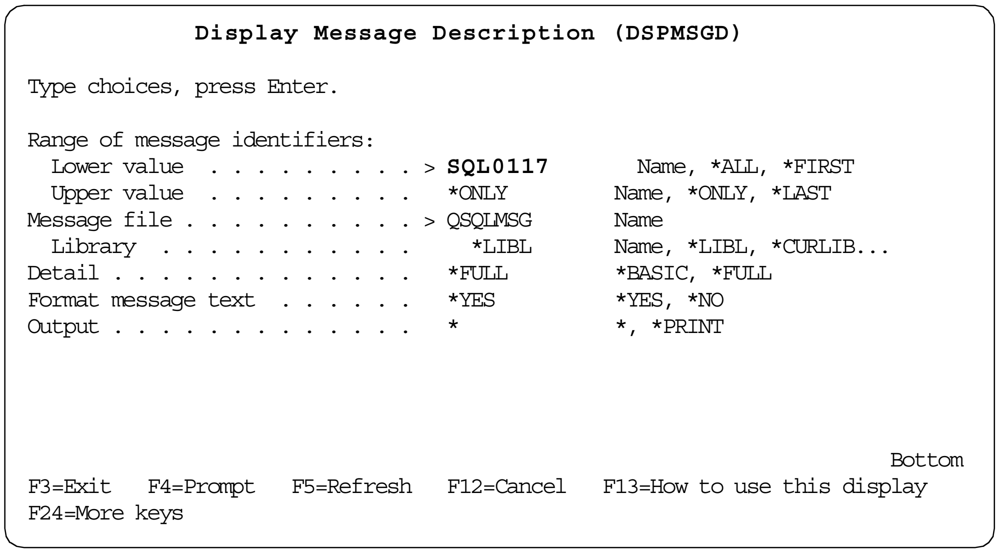
Figure 309. Log File Example

In this part of the log file, we tried to insert non-numeric price and tried to link non-existing product to an existing category. Basically, we can expect to find two types of errors:

- **Technical errors** such as non-numeric data, wrong number of columns, and so on. These errors are the result of import file with incorrect syntax.
- **Logical database errors** such as inserting a product-category relationship for non-existing category. These problems can be caused by the wrong order of the directives in the mass import input file.

Tip

If you see an SQL error number in the log file, such as 0117, you can issue the `DSPMSGD` command and find the exact error description (see Figure 310 on page 328).

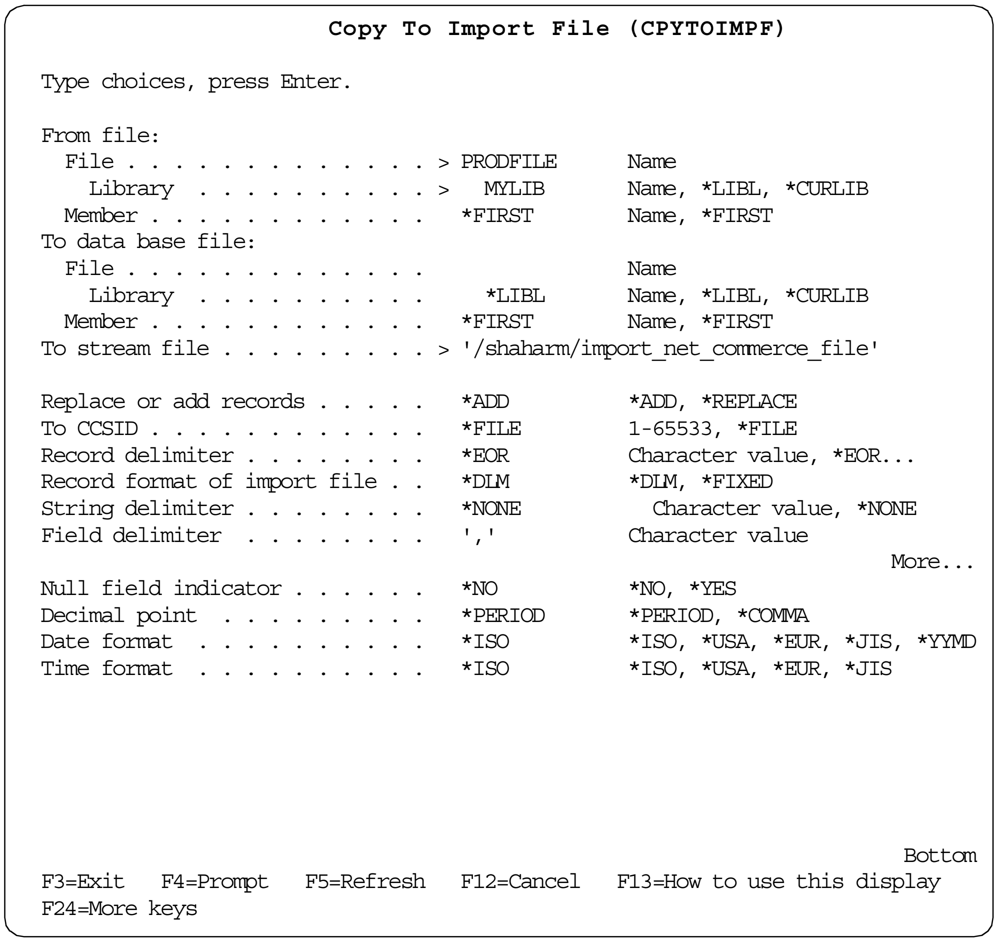


```
            Display Message Description (DSPMSGD)

Type choices, press Enter.

Range of message identifiers:
  Lower value  . . . . . . . . . > SQL0117       Name, *ALL, *FIRST
  Upper value  . . . . . . . . .   *ONLY        Name, *ONLY, *LAST
Message file . . . . . . . . . . > QSQLMSG      Name
  Library  . . . . . . . . . . .     *LIBL      Name, *LIBL, *CURLIB...
Detail . . . . . . . . . . . . .   *FULL        *BASIC, *FULL
Format message text  . . . . . .   *YES         *YES, *NO
Output . . . . . . . . . . . . .   *            *, *PRINT




                                                                  Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 310. Find Info SQL Message Description

15.2.2.4 Using the System Export Utility

The AS/400 system has an export utility that allows you to export database file to a flat file on the IFS. The export utility is invoked by the Copy to Import File (CPYTOIMPF) command. In most cases, you have to manipulate the data in the existing back-end application to prepare the import file. However, if you have a database file that is ready for importing, you can use the AS/400 export utility to create the import file. For an example, see Figure 311 on page 329.



```
                     Copy To Import File (CPYTOIMPF)

Type choices, press Enter.

From file:
  File . . . . . . . . . . . . . > PRODFILE      Name
    Library  . . . . . . . . . . >   MYLIB       Name, *LIBL, *CURLIB
  Member . . . . . . . . . . . .   *FIRST        Name, *FIRST
To data base file:
  File . . . . . . . . . . . . .                 Name
    Library  . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
  Member . . . . . . . . . . . .   *FIRST        Name, *FIRST
To stream file . . . . . . . . . > '/shaharm/import_net_commerce_file'

Replace or add records . . . . .   *ADD          *ADD, *REPLACE
To CCSID . . . . . . . . . . . .   *FILE         1-65533, *FILE
Record delimiter . . . . . . . .   *EOR          Character value, *EOR...
Record format of import file . .   *DLM          *DLM, *FIXED
String delimiter . . . . . . . .   *NONE           Character value, *NONE
Field delimiter  . . . . . . . .   ','           Character value
                                                                       More...
Null field indicator . . . . . .   *NO           *NO, *YES
Decimal point  . . . . . . . . .   *PERIOD       *PERIOD, *COMMA
Date format  . . . . . . . . . .   *ISO          *ISO, *USA, *EUR, *JIS, *YYMD
Time format  . . . . . . . . . .   *ISO          *ISO, *USA, *EUR, *JIS






                                                                        Bottom
F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
F24=More keys
```

Figure 311. AS/400 Native Export to Flat File Command Example

The export utility copies the physical file PRODFILE from library MYLIB to an IFS file named import_net_commerce_file. It contains the default column delimiter ",". The IFS file can then be imported by using the IMPNETCDAT command (*if* the original file already contained the required data for the IMPNETCDAT command).

15.2.2.5 Improving Mass Import Performance

To process records more rapidly, the Mass Import utility keeps common data, such as the parent category, in a cache for re-use with subsequent records. This saves the utility from having to query the database for each record. To benefit from the performance advantages that caching provides, you need to order records as described here:

1. Enter the records pertaining to categories in the order given in 15.2.2.1, “Import File Creation” on page 321. The first record in the category group must be #CATEGORY. The category group consists of the tables CATEGORY and CATESGP.
2. Enter the records pertaining to products as a group in the order described in 15.2.2.1, “Import File Creation” on page 321. The first record in the product group must be #PRODUCT. The product group consists of the tables PRODUCT, PRODDSTATR, PRODATR, PRODPRCS, and PRODSGP.
3. Enter child products immediately after the parent product.

4. Enter child categories immediately after the parent category.
5. Enter category product relationship records (using the `#CGPRREL` command) after the corresponding product and category groups.
6. As the sizes of the tables increase, remove any obsolete data from the tables. You can use the Database Cleanup Utility to quickly remove large amounts of data. Refer to the book *Net.Commerce for AS/400 Net.Commerce Utilities,* (nc_util.pdf) for details about the requirements for running the Database Cleanup utility.

15.3 Importing Data by Example

Our back-end system consists of product and category database tables. We developed a user-written data loading program that allows us to perform massive dataloads. Our program will also be used to take care of single row operations for the product file. It inserts and updates information in the product table in the Net.Commerce database based on operations in the back-end system product table.

15.3.1 Consideration for the Example Solution

Since we want to ensure referential integrity in the Net.Commerce database, we chose the safest approach to our solution and used the Mass Import utility. We needed some mapping between the back-end system tables to Net.Commerce tables. Therefore, we chose to write a custom routine to map the database to an import file.

15.3.2 The LOADPRD Utility — Description

The LOADPRD utility is an example utility to load data from back-end tables to Net.Commerce table. The back-end system tables contain packed fields and are typical of existing AS/400 applications.

Here is the list of files and fields that we used from the back-end system:

- ***BEPROD*** — Product table with the columns:

BEPNBR	Product number
BESDSC	Short description
BELDSC	Long description
BEPRIC	Product price
BECUR	Price currency
BEGRPC	Product group code
BEINVI	Items in inventory
BEINVC	Inventory unit code

- ***BECATEG*** — Product category table with the columns:

BECODE	Category code
BETEXT	Category description

- ***BEMEASUR*** — Unit of measure table with the columns:

MSCODE	Measurement unit code
MSTEXT	Measurement unit text

The utility creates an import file for loading the products, product prices, categories, and product-categories relationships. It also performs the following actions:

- Transforms numeric data to character data.
- Translates the measurement codes used in the legacy system to measurement text values for Net.Commerce
- Chains between products and categories
- Adds the correct Mass Import commands for: #STORE, #PRODUCT, #CATEGORY, PRODPRCS, and #CGPRREL to the output file.

We found it convenient to draw a table in which we mapped the columns needed by the Net.Commerce Mass Import utility to their equivalent back-end system sources. Table 19 displays some of the decisions we made in our example.

Table 19. Map of Mass Import Columns to Back-End System Tables

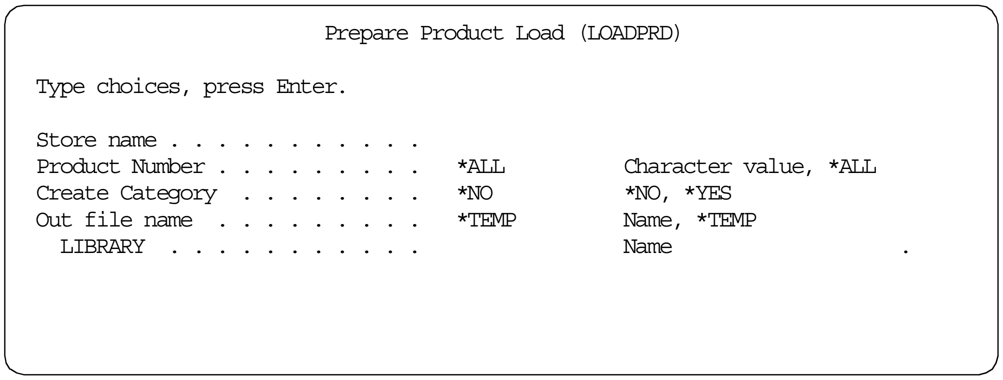
Mass Import Command and Net.Commerce Column	**Column Source in Back-end Tables**
Category name — Directive #CATEGORY column cgname	The column BETEXT in the categories table BECATEG
Product number — Directive #PRODUCT column prnbr	The Column BEPNBR in the products table BEPROD
Items in stock — Directive #PRODUCT column prvent.	Character representation of column BEINVI in table BEPROD.
Unit of measure — Directive #PRODUCT column prsmeas	The column MSTEXT in the measurement table BEMEASUR
Template macro — Directive #PRODSGP column psdisplay.	Set to constant value storename/prod1.d2w with directory name equals to the store name.
Product category — Directive #CGPRREL columns cat_name and prnbr.	Use the BEPROD product table column BEGRPC for the prnbr column and the text column BETEXT in table BECATEG for the cat_name column.
Product image file path — Directive #PRODUCT column prfull	We have no way to derive this column from our back-end system. Column remains null in the mass import file.

See the load program LOADPRDR for other mapping decisions we made.

The objects included in this utility are:

- **LOADPRD** — Load Product Data command
- **LOADPRD** — CPP program for LOADPRD command
- **LOADPRDR** — ILE RPG program which performs the actual creation of the import file
- **LOADALL** — CL program to import all products for store

Figure 312 on page 332 displays the LOADPRD prompt.



```
                    Prepare Product Load (LOADPRD)

Type choices, press Enter.

Store name . . . . . . . . . . .
Product Number . . . . . . . . .   *ALL           Character value, *ALL
Create Category  . . . . . . . .   *NO            *NO, *YES
Out file name  . . . . . . . . .   *TEMP          Name, *TEMP
  LIBRARY  . . . . . . . . . . .                  Name                   .
```

Figure 312. The LOADPRD Command Prompt

The LOADPRD command parameters are shown in Table 20.

Table 20. Command LOADPRD Parameters.

Parameter	**Parameter Description**
Store name (STORE)	The store name for which the mass import file is being created. This is a required parameter. The store name is case sensitive.
Product number (PRODUCT)	The product number that we want to load. The default is *ALL, which loads all our products. Specific existing product number that can be used to load ongoing changes in the back-end database to Net.Commerce database.
Create category(CRTCAT)	Specifies whether to create category information in the mass import file. The default is *NO. *YES can only be specified when the PRODUCT parameter value is set to *ALL
Out file name(OUTFILE)	The name of the import file to create. If this file does not exist, the command creates it. If the file exists, the command appends the new mass import directives to the existing file. The default is *TEMP, which creates the file in QTEMP by the name of #NETCIMP.

Figure 313 on page 333 shows the source to the command interface for our utility.

```
CMD        PROMPT('Prepare Product Load')

            PARM       KWD(STORE) TYPE(*CHAR) LEN(20) MIN(1) +
                         CASE(*MIXED) PROMPT('Store name')

            PARM       KWD(PRODUCT) TYPE(*CHAR) LEN(12) DFT(*ALL) +
                         SPCVAL((*ALL ' ')) PROMPT('Product Number')

            PARM       KWD(CRTCAT) TYPE(*CHAR) LEN(4) RSTD(*YES) +
                         DFT(*NO) VALUES(*NO *YES) PROMPT('Create +
                         Category')

            PARM       KWD(OUTFILE) TYPE(QUAL1) DFT(*TEMP) +
                         SNGVAL((*TEMP)) PROMPT('Out file name')
 QUAL1:     QUAL       TYPE(*NAME) LEN(10)
            QUAL       TYPE(*NAME) LEN(10) PROMPT(LIBRARY)
```

Figure 313. Command LOADPRD Source File

Figure 314 and Figure 315 on page 334 show the command processing program (CPP) for command LOADPRD.

```
*************** Beginning of data *****************************
/*********************************************************/
/* This program demonstrates the integration of back-end   */
/* system for data loading into net commerce.              */
/*                                                         */
/* Arguments:                                              */
/*            pr_store   - Store name                      */
/*            pr_prd     - Product number                  */
/*            pr_cat     - Create category ?               */
/*            pr_outf    - Out File Name                   */
/* Author: Shahar mor                                      */
/* Provided AS IS                                          */
/*********************************************************/

             PGM        PARM(&pr_store        +
                             &pr_prd          +
                             &pr_cat          +
                             &pr_outf         +
                                              )


           DCL        VAR(&Pr_store)   TYPE(*CHAR) LEN(20)
           DCL        VAR(&pr_prd  )   TYPE(*CHAR) LEN(12 )
           DCL        VAR(&pr_cat  )   TYPE(*CHAR) LEN(4 )
           DCL        VAR(&pr_outf )   TYPE(*CHAR) LEN(20)

           DCL        VAR(&parml   )   TYPE(*CHAR) LEN(37 )
           DCL        VAR(&c_file  )   TYPE(*CHAR) LEN(10 )
           DCL        VAR(&c_lib   )   TYPE(*CHAR) LEN(10 )
           DCL        VAR(&Error   )   TYPE(*CHAR) LEN(1  )  +
                                       VALUE('1')

           DCL        VAR(&msgflib)    TYPE(*CHAR) LEN(10)
           DCL        VAR(&msgf)       TYPE(*CHAR) LEN(10)
           DCL        VAR(&msgid)      TYPE(*CHAR) LEN(7)
           DCL        VAR(&msgdta)     TYPE(*CHAR) LEN(128)
```

Figure 314. The LOADPRD Command CPP Source Code (Part 1 of 2)

```
/* Set the outfile */
           CHGVAR     VAR(&c_file) VALUE(%SST(&pr_outf 1 10))
           IF         COND(&c_file = '*TEMP') THEN(DO)
             CHGVAR     VAR(&c_file) VALUE('#NETCIMP')
             CHGVAR     VAR(&c_lib ) VALUE('QTEMP')
           ENDDO
           ELSE       CMD(DO)
             CHGVAR     VAR(&c_file) VALUE(%sst(&pr_outf 1 10))
             CHGVAR     VAR(&c_lib ) VALUE(%sst(&pr_outf 11 10))
           ENDDO
/* Prepare New file                       */
             CRTPF      FILE(&c_lib/&c_file) RCDLEN(500)
             MONMSG     MSGID(CPF0000)
/* Prepare parameter for the rpg import program */
             CHGVAR     VAR(&parml) VALUE(&pr_store || &pr_prd || +
                          &pr_cat )
/* Call the File program */
             OVRDBF     FILE(OUTFILE) TOFILE(&c_lib/&c_file)
             CALL       PGM(LOADPRDR) PARM(&parml)
/* Check for result */
             IF         COND(%SST(&parml 37 1) = &error) THEN(DO)
                 CHGVAR     VAR(&msgdta) VALUE('The creation of import
                          for product:' *BCAT &pr_prd  *BCAT 'For +
                          store:' *BCAT &pr_store *BCAT 'Failed')
                 SNDPGMMSG  MSGID(CPF9897) MSGF(QCPFMSG) MSGDTA(&msgdta
                              TOPGMQ(*SAME) MSGTYPE(*ESCAPE)
             ENDDO
/* Clean up and return */
             DLTOVR     FILE(OUTFILE)
             RETURN

 STDERR:     RCVMSG     MSGTYPE(*EXCP) MSGDTA(&msgdta) +
                                       MSGID(&msgid) MSGF(&msgf) +
                                       MSGFLIB(&msgflib)

           MONMSG     MSGID(CPF0000 MCH0000)
           SNDPGMMSG  MSGID(&msgid) MSGF(&msgflib/&msgf) +
                        MSGDTA(&msgdta) MSGTYPE(*ESCAPE)
           MONMSG     MSGID(CPF0000 MCH0000)
           RETURN

           ENDPGM
```

Figure 315. The LOADPRD Command CPP Source Code (Part 2 of 2)

The following example shows the source to an ILE RPG program that will perform the transformation from our back-end database to an import file. This program implements our mapping decisions as described in Table 19 on page 331:

```
H*********************************************************************************
     H* This program is an example of creating file for mass import
     H*
     H* Input:
     H*       InStore- Store for products
     H*       InCat  - Create Category Load records ?
     H*       InProd - Product number or blank for all
     H*        IoErr  - Return error flag
     H* Author: Shahar mor
     H* Provided AS IS
   H*********************************************************************************
     H DFTACTGRP(*NO) ACTGRP(*CALLER)
     F*
     F*  Products File
     FBEPROD    IF   E           K DISK    RENAME(BEPROD:RBEPROD)
     F
     F*  Measurments unit table
     FBEMEASUR  IF   E           K DISK    RENAME(BEMEASUR:RBEMEASUR)
     F*
     F*  Products File
     FBECATEG   IF   E           K DISK    RENAME(BECATEG:RBECATEG)
     F
```

```
 F*  Ouput flat file
 FOUTFILE   O    F  500        DISK
 D*
 D*  Parameter Structure
 D PrRqs          DS
 D  InStore                    20                                      Product number
 D  InProd                     12                                      Product number
 D  InCrtc                      4                                      Product number
 D  IoErr                       1
 D*
 DLin1            S           500   Varying
 DLin2            S           500   Varying
 DLin3            S           500   Varying
 DLin4            S           500   Varying
 DLine1           DS          500
 DLine2           DS          500
 DLine3           DS          500
 DLine4           DS          500
 D*
 D*  Constants (Tailor to delimiter and Seperators Should be here)
 DSeperator       C                 ';'
 DAllProd         C                 CONST(' ')
 DDelimiter       C                 '&'
 DCatMacro        C                 '/cat1.d2w'
 DProdMacro       C                 '/prod1.d2w'
 DRoot            C                 'Top Category'
 DSeqnbr          C                 '1'
 D*
 DSetNulls        PR            50A
 D NbrNulls                      3S 0 VALUE
C*********************************************************************************
 C*                   Main logic
C*********************************************************************************
 C*
 C     *ENTRY        PLIST
 C                   PARM                    PrRqs
 C
 C*
 C* Check for product existence(Single record approach)
 C                   EVAL      IoErr = *OFF
 C                   IF        InProd <> AllProd
 C     InProd        CHAIN(E)  RBEPROD
 C                   IF        Not %found
 C                   MOVE      *ON           IoErr
 C                   ENDIF
 C                   ENDIF
 C*
 C*   Perform The loading
 C                   IF        IoErr = *OFF
 C                   EXSR      PrpHdr
 C*
 C* Should we prepare category Records ?
 C                   IF        InCrtc  = '*YES'
 C                   EXSR      PrpCateg
 C                   ENDIF
 C*
 C                   IF        InProd <> AllProd
 C                   EXSR      PrpProd
 C                   ELSE
 C                   EXSR      PrpAll
 C                   ENDIF
 C                   ENDIF
 C*
 C                   EVAL      *Inlr = *ON
 C*******************************************************************
 C     PrpCateg      BEGSR
 C*******************************************************************
 C*
 C*  Prepare Categories records for mass import.
 C*  Our shop contains only one level of categories.
 C*  Our shop will use the category description as the category name
 C*
 C     *LOVAL        SETLL     RBECATEG
 C                   READ(E)   RBECATEG
 C*
 C                   DOW       Not %EOF(BECATEG)
 C                   EVAL      Lin1 = '#CATEGORY' + Seperator
 C                   EVAL      Lin1 = Lin1 + %trim(BETEXT) + Seperator     Category name
```

```
C                 EVAL      Lin1 = Lin1 + %trim(BETEXT) + Seperator      Short description
C                 EVAL      Lin1 = Lin1 + %trim(BETEXT) + Seperator      Long description
C                 EVAL      Lin1 = Lin1 + %trim(SetNulls(6))
C                 EVAL      Lin1 = Lin1 + %trim(Root)                    Long description
C                 EVAL      Lin1 = Lin1 + Seperator
C                 EVAL      Lin1 = Lin1 + Seqnbr                         Long description
C                 EVAL      Lin1 = Lin1 + Delimiter
C                 EVAL      Line1 = %subst(Lin1:1:%Len(Lin1))
C                 WRITE     OUTFILE       Line1
C                 EVAL      Lin1 = '#CATESGP' + Seperator
C                 EVAL      Lin1 = Lin1 + %trim(BETEXT) + Seperator
C                 EVAL      Lin1 = Lin1 + %trim(SetNulls(1 ))
C                 EVAL      Lin1 = Lin1 + %trim(InStore )
C                 EVAL      Lin1 = Lin1 + %trim(CatMacro)
C                 EVAL      Lin1 = Lin1 + %trim(SetNulls(3 ))
C                 EVAL      Lin1 = Lin1 + Delimiter
C                 EVAL      Line1 = %subst(Lin1:1:%Len(Lin1))
C                 WRITE     OUTFILE       Line1
C                 READ(E)   RBECATEG
C                 ENDDO
C*
C                 ENDSR
C*******************************************************************
C     PrpHdr        BEGSR
C*******************************************************************
C*
C*  Prepare Header with store information
C*
C                 EVAL      Lin1 = '#ROWDELIMITER;'
C                 EVAL      Lin1 = Lin1 + Delimiter
C                 EVAL      Lin1 = Lin1 + Delimiter
C                 EVAL      Line1 = %subst(Lin1:1:%Len(Lin1))
C                 WRITE     OUTFILE       Line1
C*
C                 EVAL      Lin1 = '#STORE;'
C                 EVAL      Lin1 = Lin1 + %trim(InStore)
C                 EVAL      Lin1 = Lin1 + Delimiter
C                 EVAL      Line1 = %subst(Lin1:1:%Len(Lin1))
C                 WRITE     OUTFILE       Line1
C*
C                 ENDSR
C*******************************************************************
C     PrpAll        BEGSR
C*******************************************************************
C*
C*  Prepare all products to import file
C*
C     *LOVAL        SETLL     RBEPROD
C                 READ(E)   RBEPROD
C*
C                 DOW       Not %EOF(BEPROD)
C                 EXSR      PrpProd
C                 READ(E)   RBEPROD
C                 ENDDO
C*
C                 ENDSR
C*******************************************************************
C     PrpProd       BEGSR
C*******************************************************************
C*
C*  Prepare current record to import file
C*
C                 EVAL      Lin1 = '#PRODUCT;'
C                 EVAL      Lin2 = '#PRODPRCS;'
C                 EVAL      Lin3 = '#PRODSGP;'
C                 EVAL      Lin4 = '#CGPRREL;'
C*
C******************************************************************
C* Setting the General Product information (See #PRODUCT on Doc)
C*
C* Every Seperator(;) Is representing null field. It Will be inserted
C* To the net commerce database with default values.
C*
C* You could perform some more data mapping in this stage. You could
C* Also Insert some different default data. For example, In The Product
C* Image file field(PRFULL) We could insert default path with the
C* product number as the file name.
C******************************************************************
```

```
    C*
    C                   EVAL      Lin1 = Lin1 + %trim(BEPNBR)                 Product Number
    C                   EVAL      Lin1 = Lin1 + Seperator                     Null parent
    C                   EVAL      Lin1 = Lin1 + Seperator +%trim(BESDSC)      Short desc.
    C                   EVAL      Lin1 = Lin1 + Seperator + %trim(BELDSC)     Short desc.
    C                   EVAL      Lin1 = Lin1 + %trim(SetNulls(11))
    C*
    C* Need to translate the legacy code to net commerce text code
    C     BEINVC        CHAIN(E)  RBEMEASUR
    C                   IF        %Found
    C                   EVAL      Lin1 = Lin1 + %trim(MSTEXT)
    C                   ELSE
    C                   EVAL      Lin1 = Lin1 + Seperator
    C                   ENDIF
    C*
    C                   EVAL      Lin1 = Lin1 + %trim(SetNulls(4))
    C                   EVAL      Lin1 = Lin1 + %trim(%editc(BEINVI:'Z'))     Product Number
    C                   EVAL      Lin1 = Lin1 + %trim(SetNulls(10))
    C                   EVAL      Lin1 = Lin1 + Delimiter                     Null Fields
    C                   EVAL      Line1 = %subst(Lin1:1:%Len(Lin1))
    C* Setting the Price information(See #PRDPRCS on Documantation)
    C                   EVAL      Lin2 = Lin2 + %trim(BEPNBR)                 Product Number
    C                   EVAL      Lin2 = Lin2 +   Seperator + Seperator
    C                   EVAL      Lin2 = Lin2 + %trim(%editc(BEPRIC:'3'))     Product Price
    C                   EVAL      Lin2 = Lin2 + Seperator
    C                   EVAL      Lin2 = Lin2 + BECUR                         Product Currency
    C                   EVAL      Lin2 = Lin2 + %trim(SetNulls(4))
    C                   EVAL      Lin2 = Lin2 + Delimiter                     Null Fields
    C                   EVAL      Line2 = %subst(Lin2:1:%Len(Lin2))
    C* Setting the Product template (See #PRODSGP on Documantation)
    C                   EVAL      Lin3 = Lin3 + %trim(BEPNBR)                 Product Number
    C                   EVAL      Lin3 = Lin3 +   Seperator + Seperator
    C                   EVAL      Lin3 = Lin3 + %trim(InStore)
    C                   EVAL      Lin3 = Lin3 + %trim(ProdMacro)
    C                   EVAL      Lin3 = Lin3 + %trim(SetNulls(3))
    C                   EVAL      Lin3 = Lin3 + Delimiter                     Null Fields
    C                   EVAL      Line3 = %subst(Lin3:1:%Len(Lin3))
    C*
    C* Setting the Category information for this product
    C*
    C                   WRITE     OUTFILE       Line1
    C                   WRITE     OUTFILE       Line2
    C                   WRITE     OUTFILE       Line3
    C*
B001C                   IF        InCrtc = '*YES'
    C     BEGRPC        CHAIN(E)  RBECATEG
    C                   IF        %Found
    C                   EVAL      Lin4 = Lin4 + %trim(BETEXT)                 Category
    C                   EVAL      Lin4 = Lin4 + Seperator
    C                   EVAL      Lin4 = Lin4 + %trim(BEPNBR)                 Product Number
    C                   EVAL      Lin4 = Lin4 + Seperator
    C                   EVAL      Lin4 = Lin4 + Delimiter                     Null Fields
    C                   EVAL      Line4 = %subst(Lin4:1:%Len(Lin4))
    C                   WRITE     OUTFILE       Line4
E001C                   ENDIF
    C                   ENDIF
    C*
    C                   ENDSR
   C*******************************************************************
    PSetNulls         B
    DSetNulls         PI            50A
    D NbrNulls                       3S 0 VALUE
    DNullFld          S             50
    C*
    C*  Will Set seperators to Number of null Field required
    C*
    C                   EVAL      NullFld = *BLANK
    C                   DO        NbrNulls
    C                   CAT       Seperator:0   NullFld
    C                   ENDDO
    C*
    C                   Return    NullFld
    C*
    PSetNulls         E
```

Note

- Pay attention to the usage of the %EDITW that converts numbers to formatted text.
- Use the %TRIM function to avoid long blanks in the input.
- The + sign is used to concatenate strings instead of the CAT operation.
- The output file is program described file in the QSYS.LIB file system. You may use the `DSPPFM` command to display its contents. You may find it convenient to copy the file to the IFS root file system by using the `CPYTOSTMF` command.
- The SetNull internal function is used to insert place holders for columns defined by the mass import directives, but we have no way of supplying them.

We can define a process in which mass import file creation and the Mass Import utility will run in one step. The process will convert our back-end database to Net.Commerce tables.

Figure 316 on page 339 shows an example of the CL program LOADALL that uses the LOADPRD command with Net.Commerce Mass Import utility to load the back-end data into the Net.Commerce database.

```
/*********************************************************/
/* This program will load data from back-end system to     */
/* Net.Commerce database.                                  */
/*                                                         */
/* Author: Shahar mor                                      */
/* Provided AS IS                                          */
/*********************************************************/
             PGM

/******************************************************************/
/* Phase 1 - Prepare the requested file . This phase will create    */
/* Import file for store 'NetAway' with all categories and products */
/*                                                                  */
/* The import file will be located in qtemp and will be named       */
/* #NETCIMP.                                                        */
/******************************************************************/

             LOADPRD    STORE(NetAway ) CRTCAT(*YES)
/******************************************************************/
/* Phase 2 - Use mass import to populate data to the Net.Commerce   */
/* database. Mass import with the following parameters:             */
/*                                                                  */
/*   Instance = The name of the Net.Commerce instance(TEST)         */
/*   Passwd = The password of the user profile who ownes the schema.*/
/*            In our example it is the password of usrprf TEST      */
/*    Infile = The import file we created with LOADPRD              */
/*    Database = The AS/400 local database(AS01 in our example)     */
/*    Log = The path of the log file to keep the mass import results*/
/*          (We placed this file in library loglib. The log file    */
/*           is program defined table)                              */
/*    CmtCount = The commit count. In order to improve the perform. */
/*               of the import we chose to commit only every 10     */
/*               transactions                                       */
/******************************************************************/

           CRTPF      FILE(LOGLIB/LOGFILE) RCDLEN(500) TEXT('Mass +
                        import log file')
           IMPNETCDAT INSTANCE(TEST) PASSWD(PWTEST) +
                        INFILE('/qsys.lib/qtemp.lib/#netcimp.file/#+
                        netcimp.mbr') DATABASE(AS01) +
                        LOG('/qsys.lib/loglib.lib/logfile.file/logf+
                        ile.mbr')
           ENDPGM
```

Figure 316. LOADALL Utility Source Code

15.3.3 Ongoing Synchronization of Database Activity

In our example, we use the LOADPRD utility to keep synchronization between our back-end products table to Net.Commerce tables. We attach a trigger program to our existing back-end products table. Whenever a product is inserted or updated, our trigger program is called and uses LOADPRD to create the import file for specific changed product. The import file is imported to the Net.Commerce database once in a while using the Mass Import utility.

To implement our on-going synchronization approach, we must first write the trigger program. Figure 317 and Figure 318 on the following pages show our trigger program BEPRODT source code.

```
H**********************************************************************
     H*  This trigger program will be called after insert/update operation *
     H*  to the back-end system product table. The changed/new record will *
     H*  be written to an import file using the LOADPRD utility           *
     H*                                                                    *
     H*  The import file will be proccesed at later time by the           *
     H*  Net.Commerce mass import utility.                                 *
     H*  Author: Shahar mor                                                *
     H* Provided AS IS                                                     *
     H*********************************************************************
     D*
     D*  The General buffer for trigger programs
     D*
     D Parm1           DS
     D Filenm                  1     10
     D Libnm                  11     20
     D Mbrnam                 21     30
     D Trgevn                 31     31
     D Trgtim                 32     32
     D Trglvl                 33     33
     D Trgreserv              34     36
     D Trgccsid               37     40B 0
     D Trgreserv2             41     48
     D Oldofs                 49     52B 0
     D Oldlen                 53     56B 0
     D Omapof                 57     60B 0
     D Omapln                 61     64B 0
     D Newofs                 65     68B 0
     D Newlen                 69     72B 0
     D Nmapof                 73     76B 0
     D Nmapln                 77     80B 0
     D Trgreserv3             81     96
     D*
     D* Pointers for record reference
     D  Intarr         S              1A   based(intptr) dim(32767)
     D  Intptr         S               *
     D*
     D Parm2           ds
     D Len                            9b 0

     D Bimage          S               *
     D Aimage          S               *

     D Bfile         E DS                  EXTNAME(BEPROD)
     D                                     BASED(Bimage)
     D                                     PREFIX(B)
     D Afile         E DS                  EXTNAME(BEPROD)
     D                                     BASED(Aimage)
     D                                     PREFIX(A)
     D*
     D UpdNetc         PR                  EXTPGM('QCMDEXC')
     D   Cmd                       3000A   OPTIONS(*VARSIZE) CONST
     D   Cmdlen                      15P 5 CONST
     D*
     DCmdStr           S           1000    VARYING
    DStoreName       C                 'ShopITSO'
   D*************************************************************
    D*                 ***   MAIN   ***                       *
   D*************************************************************
    C
 001 C                 EXSR      Init
    C                 EXSR      LoadPrd
    C
    C                 RETURN
```

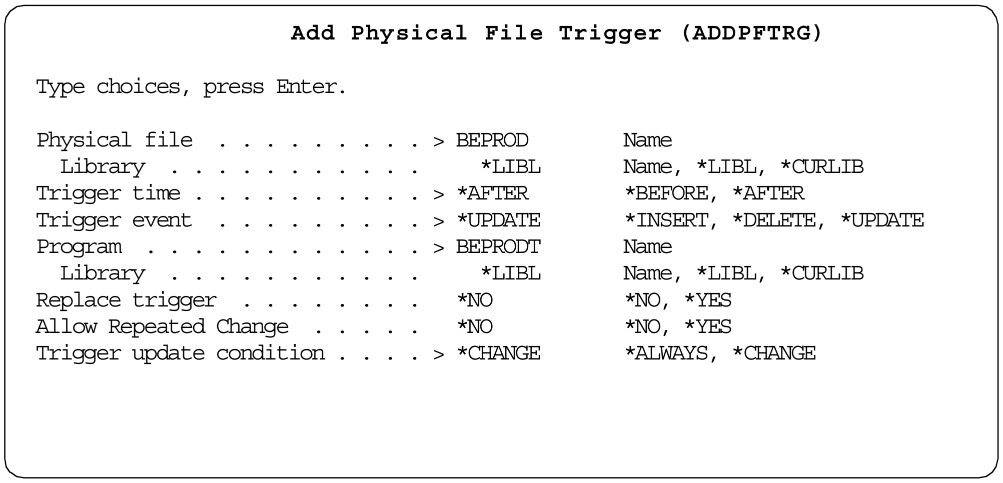
Figure 317. BEPRODT Source File (Part 1 of 2)

```
C*************************************************************
  C    LoadPrd      begsr
 C*************************************************************
  C*
 C* Prepare string for LOADPRD into temporary file. In our example we
  C* Use constant store name.
  C*
  C               EVAL     CmdStr = 'LOADPRD STORE(' + StoreName + ')' +
  C                                  ' PRODUCT(' +
  C                                 %trim(ABEPNBR) + ')' +
  C                                  ' OUTFILE(NETCBE/PRODTEMP)'
  C                CALLP    UpdNetc(CmdStr:
  C                              %len (CmdStr))
  C*
  C*
 C* In case of price change clear the cach. we use constant merchant
  C                IF       BBEPRIC <> ABEPRIC
  C               EVAL     CmdStr = 'CLRCACH MERCHANT(6)' +
  C                                  ' PRODUCT(' +
  C                                 %trim(ABEPNBR) + ')'
  C                CALLP    UpdNetc(CmdStr:
  C                              %len (CmdStr))
  C                ENDIF
  C*
  C                ENDSR
 C*************************************************************
  C    init         begsr
 C*************************************************************
  C    *entry       plist
  C                parm                 parm1
  C                parm                 parm2

  C                eval     intptr = %addr(parm1)
  C               eval     bimage = %addr(intarr(oldofs+1))
  C               eval     aimage = %addr(intarr(newofs+1))



   C                   endsr
```

Figure 318. BEPRODT Source File (Part 2 of 2)

The program is called when a database event occurs and receives parameters from the database manager. It maps the parameters to the BEPROD table structure and calls the LOADPRD command to create the import file directives in a temporary table called PRODTEMP. The trigger program also clears the Net.Commerce cache. This process is described in Chapter 17, “Interfacing to Our Back-End Business System” on page 383. Use the `CRTBNDRPG` command to create the trigger program.

After successful compilation of the BEPRODT program, we can now use it as a trigger program for update and insert operations. Figure 319 on page 342 shows an example for adding the update trigger event to the product file BEPROD in the library NETCBE. The trigger is fired after an update event only if the record was changed. The before and after images are different.



```
                    Add Physical File Trigger (ADDPFTRG)

 Type choices, press Enter.

 Physical file  . . . . . . . . . > BEPROD        Name
   Library  . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
 Trigger time . . . . . . . . . . > *AFTER        *BEFORE, *AFTER
 Trigger event  . . . . . . . . . > *UPDATE       *INSERT, *DELETE, *UPDATE
 Program  . . . . . . . . . . . . > BEPRODT       Name
   Library  . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
 Replace trigger  . . . . . . . .   *NO           *NO, *YES
 Allow Repeated Change  . . . . .   *NO           *NO, *YES
 Trigger update condition . . . . > *CHANGE       *ALWAYS, *CHANGE
```

Figure 319. Add Physical File Trigger (ADDPFTRG) Display

Important

The trigger program shown in Figure 319 was not optimized for daily operation. We include the program just to show you some capabilities of day-to-day synchronization using triggers and mass import of one row at-a-time.

Chapter 16. Setting Up Payment Methods

This chapter contains important information that you must know in order to use the different payment methods. Before you install and configure the Payment Server, you must read Chapter 6, “Planning: Payment Collection” on page 91.

16.1 Secure Electronic Transaction

Secure Electronic Transaction (SET) is an open-network, payment-card protocol. It provides greater confidentiality, greater transaction integrity, and less opportunity for fraud at all transaction points than any other existing secure payment system. The process involves a series of security checks performed using digital certificates, which are issued to participating purchasers, merchants, banks, and payment brands.

SET has four components:

- A **Cardholder Wallet component** that is run by an online consumer enabling secure payment card transactions over a network. SET Cardholder Wallet components must generate SET protocol messages that can be accepted by SET Merchant, Payment Gateway, and Certificate Authority components.
- A **Merchant Server component** that is run by an online merchant to process payment card transactions and authorizations. It communicates with the Cardholder Wallet, Payment Gateway, and Certificate Authority components.
- A **Payment Gateway component** that is run by an acquirer or a designated third party that processes merchant authorization and payment messages (including payment instructions from cardholders) and interfaces with private financial networks.
- A **Certificate Authority component** that is run by a Certificate Authority that is authorized to issue and verify digital certificates as requested by Cardholder Wallet components, Merchant Server components, or Payment Gateway components over public and private networks.

For more information about SET, go to the Web site at: `http://www.setco.org`

16.1.1 Installing Payment Server

The product is installed in the standard manner using the Restore LIC Program (RSTLICPGM) command. The objects created during installation are:

- QPYMSVR library (the library contains programs, service programs, commands, a message file, and a header file)
- QPYMSVR user profile
- /QIBM/ProdData/HTTP/Protect/PymSvr directory and subdirectories
- /QIBM/ProdData/PymSvr directory and subdirectories

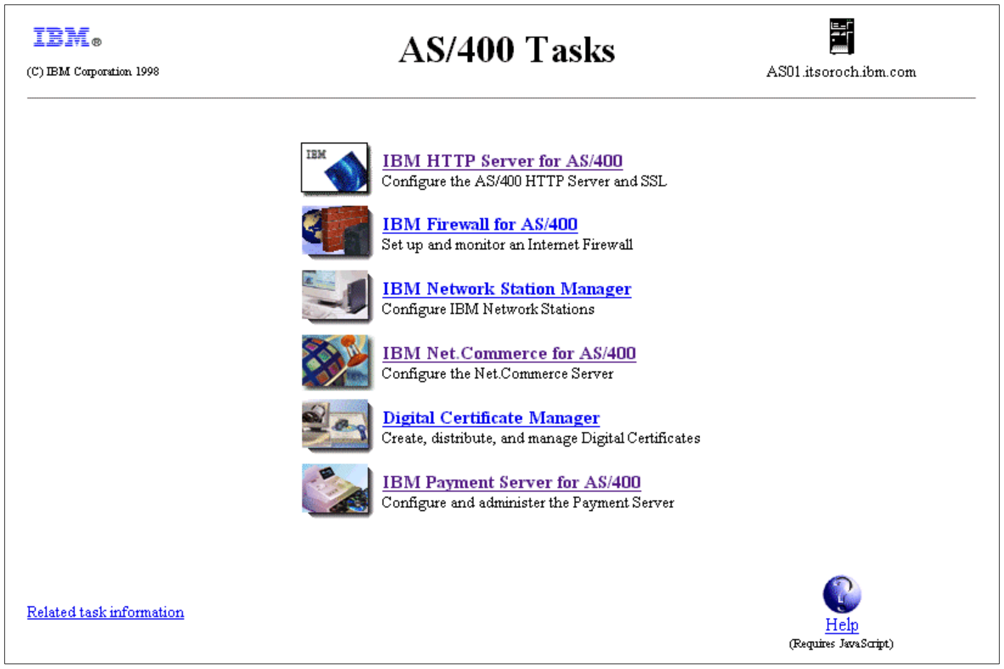
The product can be deleted by using the Delete LIC Program (DLTLICPGM) command. The QPYMSVR user profile is not deleted.

16.1.2 Creating a Payment Server

You have to create a Payment Server before you can request a digital certificate from a CA. The Payment Server has to be created before the Net.Commerce server if you want to select the use of the Payment Server with your Net.Commerce site. You may install the Payment Server later and then go back to the Net.Commerce configuration and change the setting. It is easier to add the Payment Server during the initial setup of the Net.Commerce instance.

When you choose to use a CA to issue a server certificate, you must first request the certificate. To do so, follow these steps:

1. From the AS/400 Task page (Figure 320), click **IBM Payment Server for AS/400.**



IBM®

(C) IBM Corporation 1998

AS/400 Tasks

AS01.itsoroch.ibm.com

IBM HTTP Server for AS/400
Configure the AS/400 HTTP Server and SSL

IBM Firewall for AS/400
Set up and monitor an Internet Firewall

IBM Network Station Manager
Configure IBM Network Stations

IBM Net.Commerce for AS/400
Configure the Net.Commerce Server

Digital Certificate Manager
Create, distribute, and manage Digital Certificates

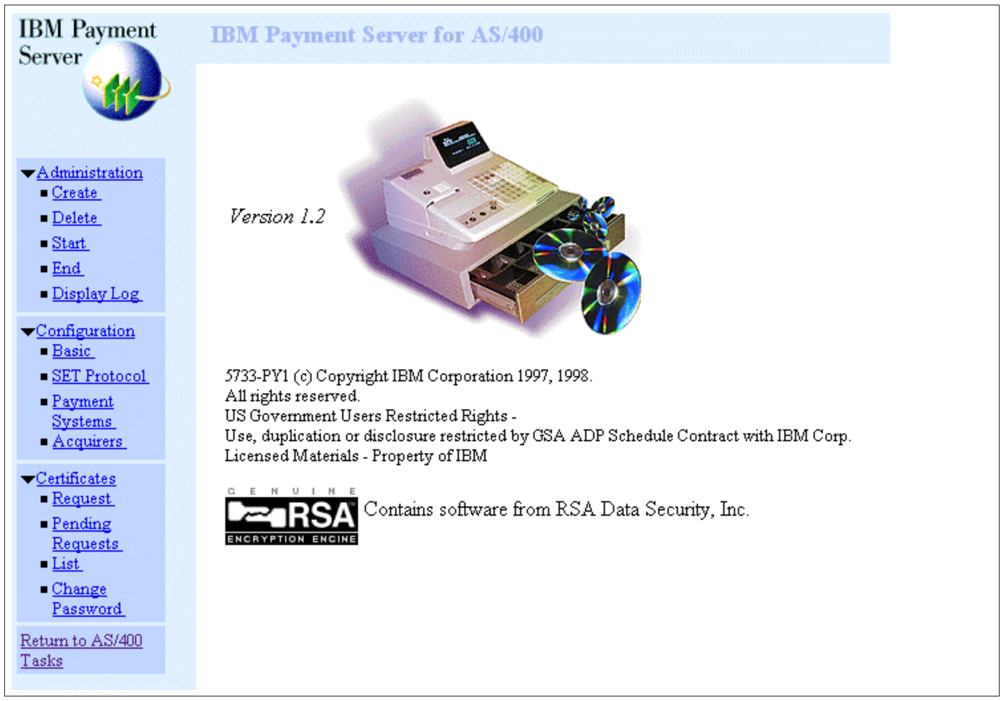
IBM Payment Server for AS/400
Configure and administer the Payment Server

Related task information

Help
(Requires JavaScript)

Figure 320. AS/400 Task Page

2. On the IBM Payment Server for AS/400 page, click on **Administration** in the left-hand frame to display an extended list of server tasks. This takes you to the display shown in Figure 321 on page 345.



5733-PY1 (c) Copyright IBM Corporation 1997, 1998.
All rights reserved.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Licensed Materials - Property of IBM

Contains software from RSA Data Security, Inc.

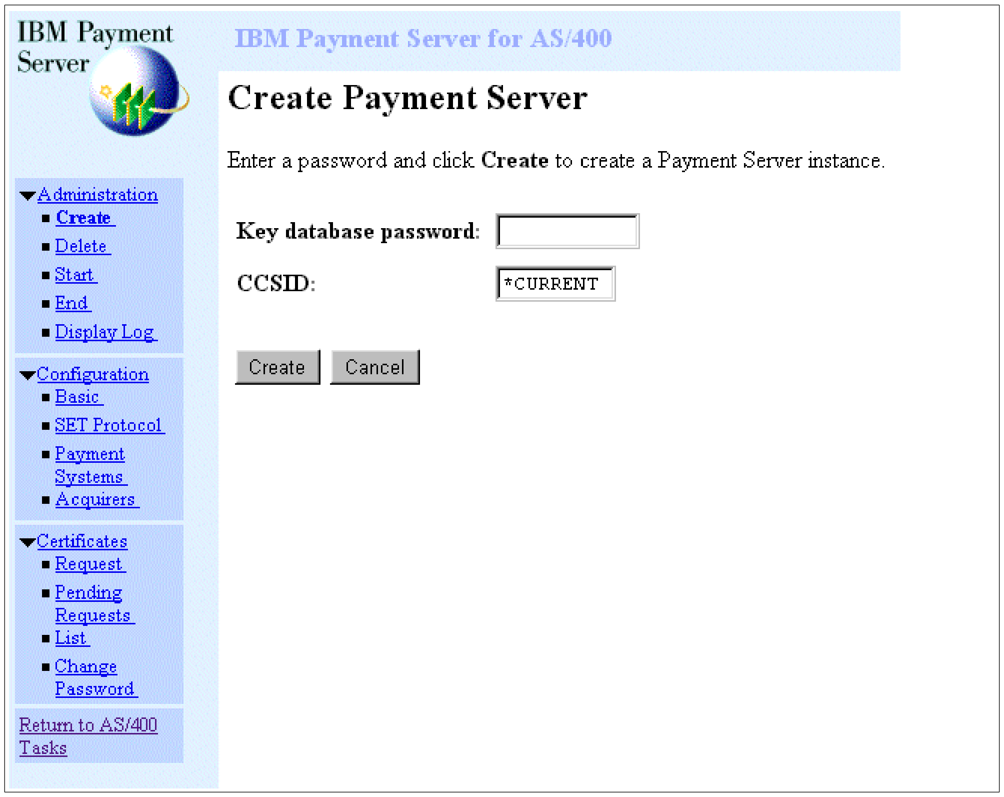
Figure 321. Payment Server for AS/400

3. Click on **Create** to go to the Create Payment Server page.
4. On the Create Payment Server page, type in a Key database password (Figure 322 on page 346), and click **Create**.

Important

If you lose your password, there is no way to recover your certificates. You will need to request them again. This is different than other AS/400 passwords, where the security officer can reset the password for you and nothing is lost. We recommend that you back up your data just prior to changing the password.

Write down the password, and save it on a secure place. You will need it when you request a digital certificate for your SET Merchant, or every time you start your Payment Server.



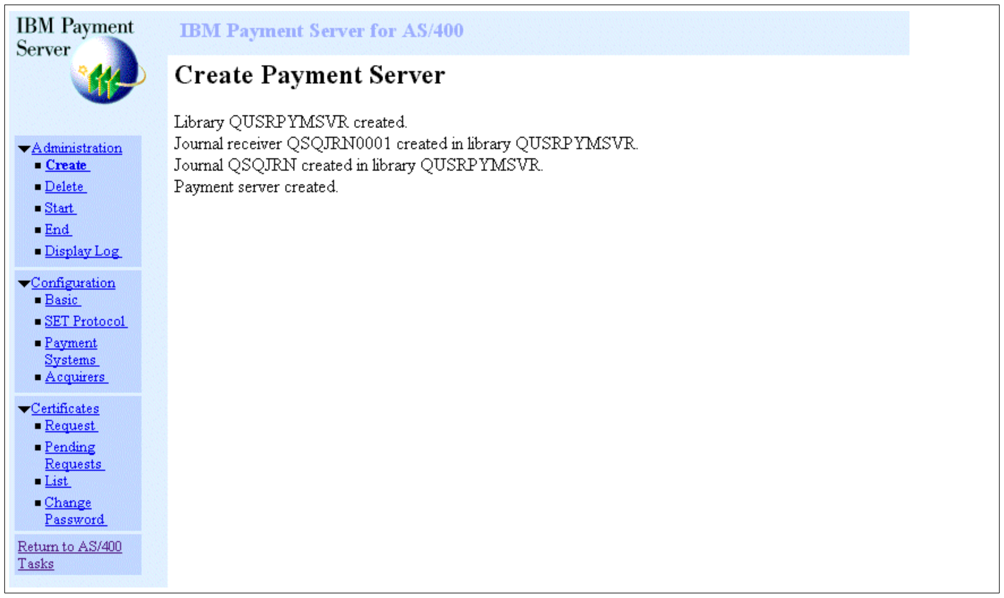
Create Payment Server

Enter a password and click **Create** to create a Payment Server instance.

Key database password:	
CCSID:	*CURRENT

Figure 322. Payment Server Administration Create Page

The Payment Server is created (Figure 323).



Create Payment Server

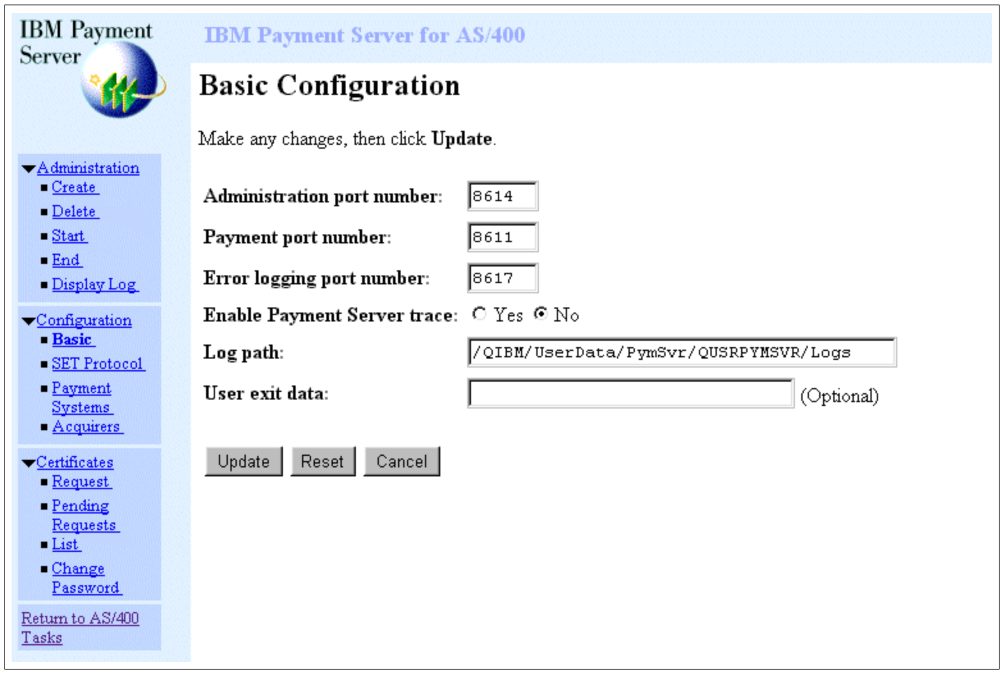
```
Library QUSRPYMSVR created.
Journal receiver QSQJRN0001 created in library QUSRPYMSVR.
Journal QSQJRN created in library QUSRPYMSVR.
Payment server created.
```

Figure 323. Payment Server Administration Creation Done Page

16.1.3 Basic Configuration of the Payment Server

To use your Payment Server, you have to complete its basic configuration. To do this, follow these steps:

1. From the IBM Payment Server for AS/400 page, click **Configuration** in the left-hand frame to display an extended list of server tasks.
2. Click on **Basic** from the list to perform the basic configuration.
3. On the Basic Configuration page (Figure 324), only make changes if your acquirer has told you to do so.



Basic Configuration

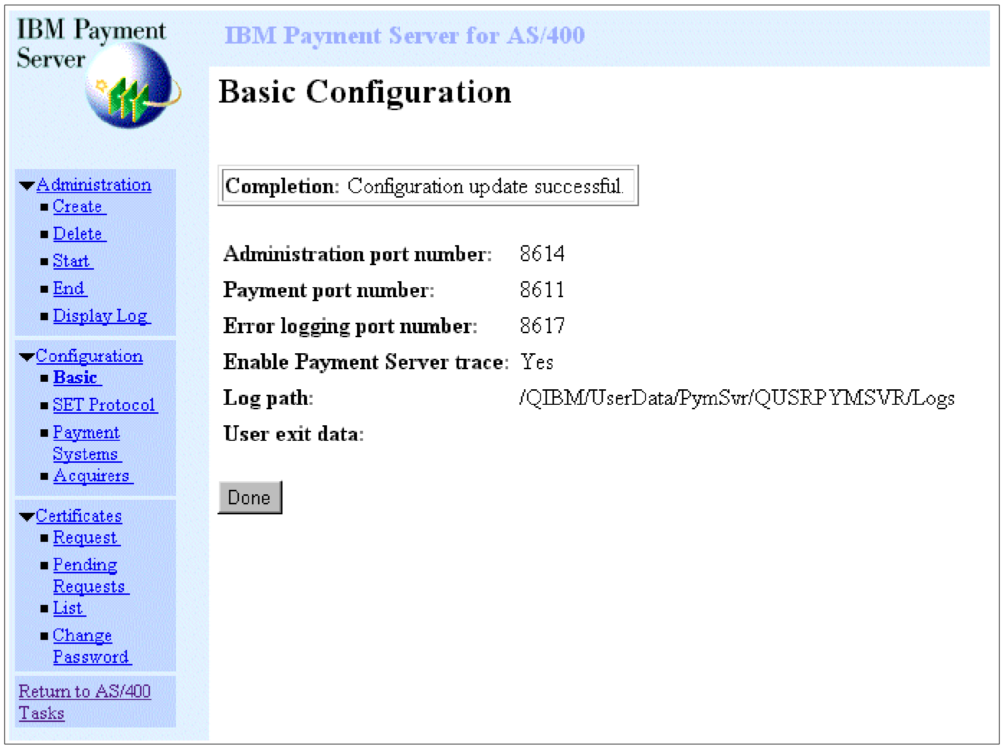
Make any changes, then click **Update**.

Administration port number:	8614
Payment port number:	8611
Error logging port number:	8617
Enable Payment Server trace:	Yes [✓ No]
Log path:	/QIBM/UserData/PymSvr/QUSRPYMSVR/Logs
User exit data:	(Optional)

Figure 324. Payment Server Configuration Basic Page

Click **Update**.

You receive a message, which indicates that the configuration update is successful (Figure 325 on page 348).



Basic Configuration

Completion: Configuration update successful.

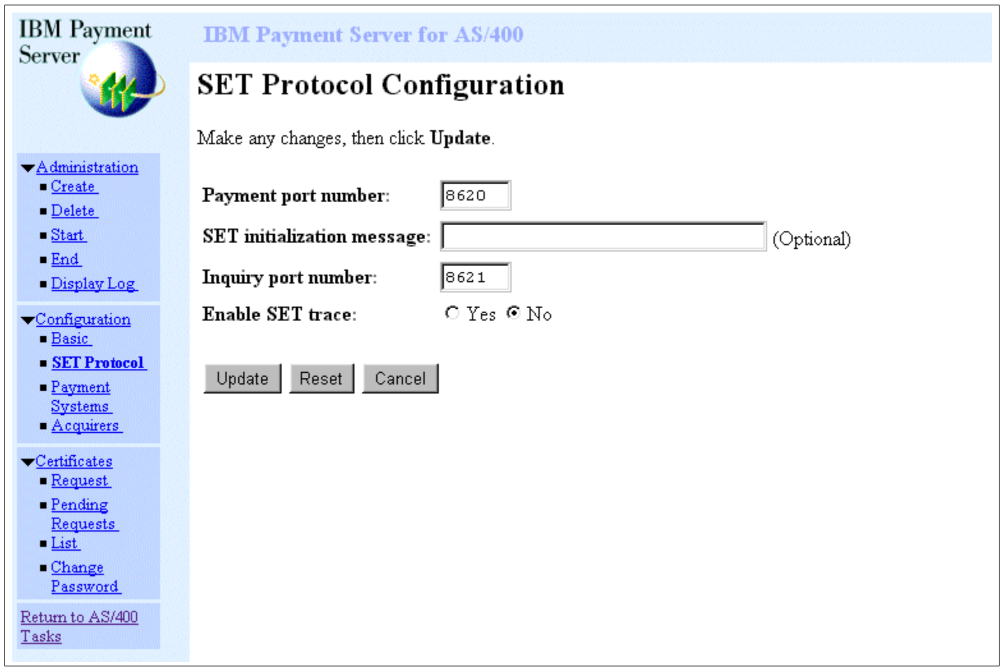
Administration port number:	8614
Payment port number:	8611
Error logging port number:	8617
Enable Payment Server trace:	Yes
Log path:	/QIBM/UserData/PymSvr/QUSRPYMSVR/Logs
User exit data:	

Figure 325. Payment Server Basic Configuration Complete Page

16.1.4 SET Protocol Configuration of the Payment Server

The next step in the configuration of the Payment Server is to configure the SET Protocol. Complete the following steps:

1. Click on **SET Protocol** from the list to perform the SET Protocol configuration.
2. On the SET Protocol Configuration page (Figure 326 on page 349), only make changes if your acquirer tells you to make them.



SET Protocol Configuration

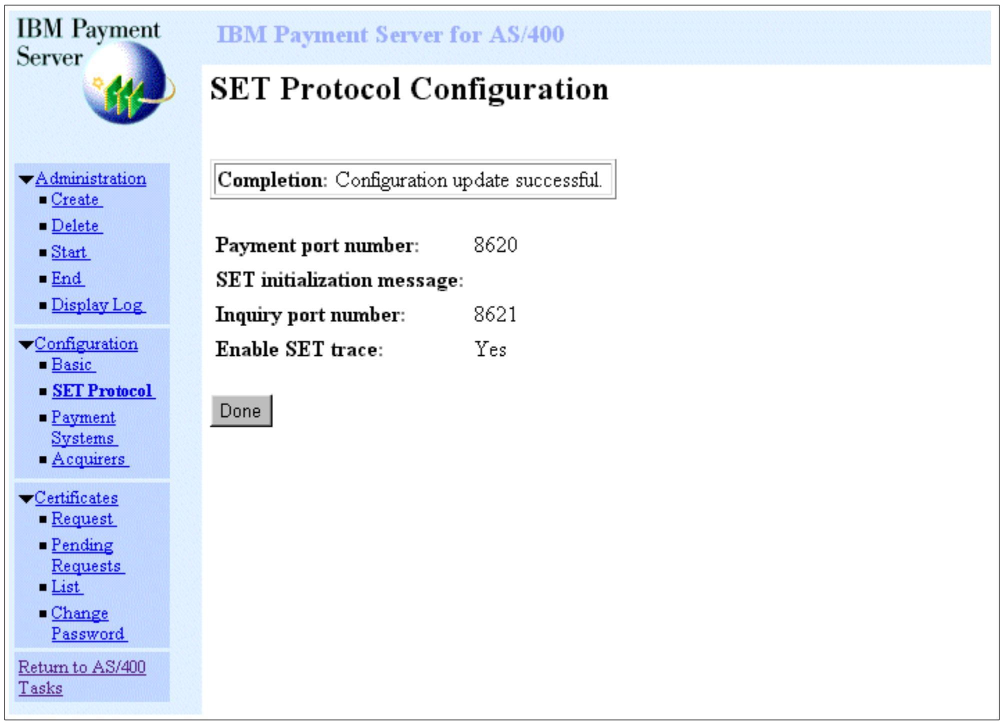
Make any changes, then click **Update**.

Payment port number:	8620	
SET initialization message:		(Optional)
Inquiry port number:	8621	
Enable SET trace:	Yes [✓ No]	

Figure 326. Payment Server SET Protocol Configuration Page

Click **Update**.

You receive a message, which indicates that the configuration update is successful (Figure 327).



SET Protocol Configuration

Completion: Configuration update successful.

Payment port number:	8620
SET initialization message:	
Inquiry port number:	8621
Enable SET trace:	Yes

Figure 327. Payment Server SET Protocol Configuration Complete Page

16.1.5 Payment Systems Configuration of the Payment Server

The next step in configuring the Payment Server is to configure the Payment System. This is done using the Net.Commerce Administrator page. To use your Payment Server with your Net.Commerce server, you have to configure both the Net.Commerce server and the Payment Server. To do so, follow this process:

1. From your administration PC browser, open your Net.Commerce Administrator page, (`http://<host_name.net>/ncadmin/`), and logon (Figure 328).

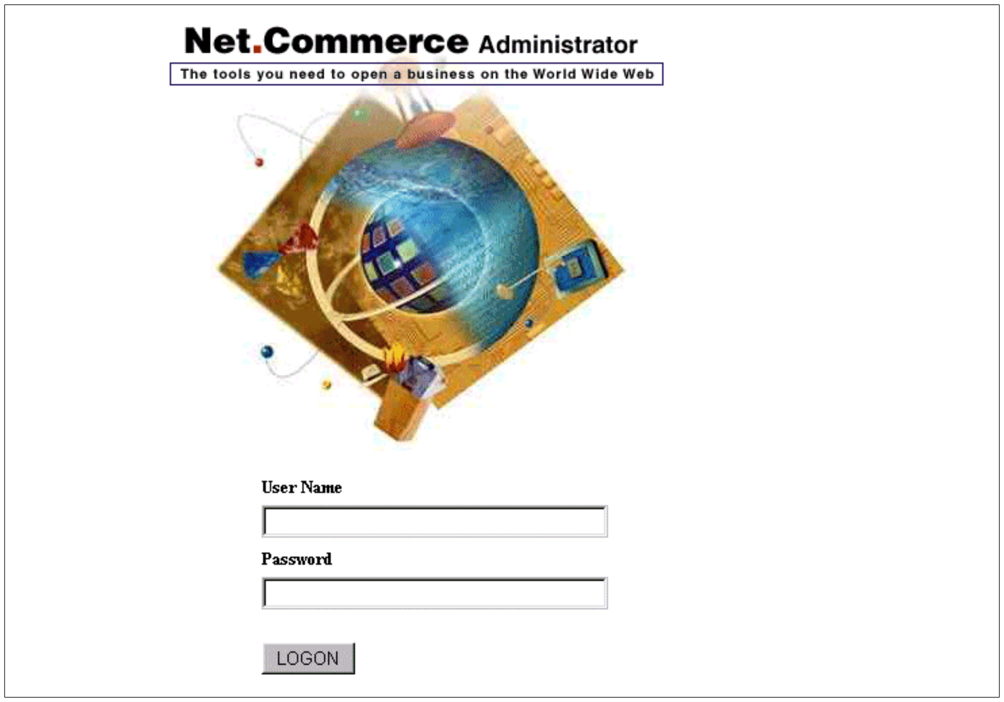
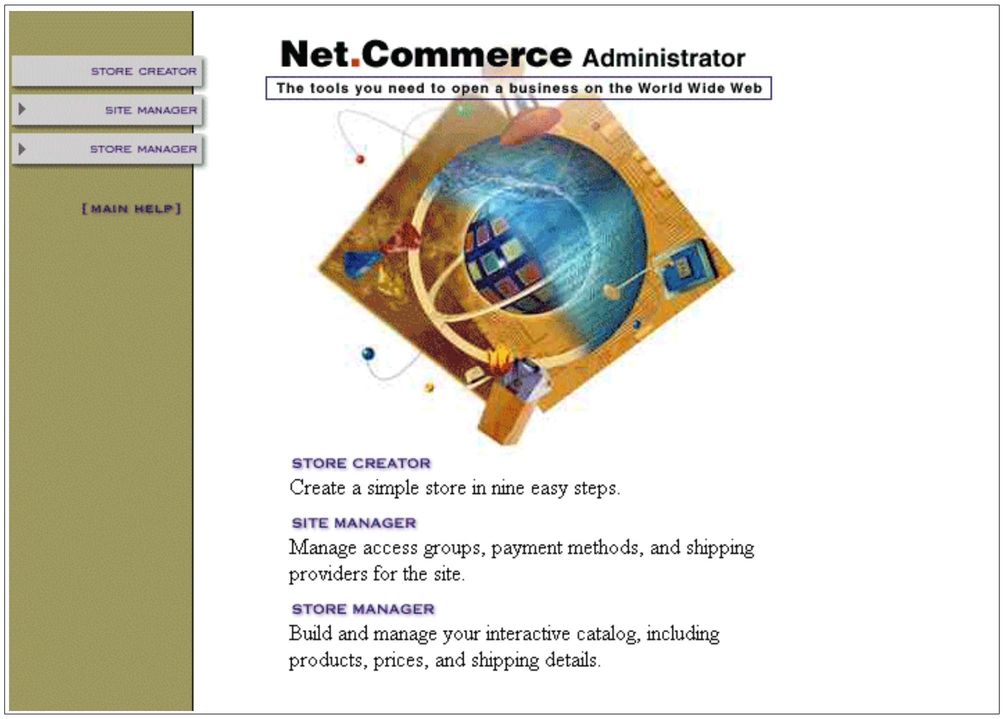

Net.Commerce Administrator
The tools you need to open a business on the World Wide Web
User Name
Password
LOGON

Figure 328. Net.Commerce Administrator Logon Page

2. From the Net.Commerce Administrator page (Figure 329 on page 351), click **Store Manager** in the left-hand frame to display an extended list of server tasks.


STORE CREATOR
SITE MANAGER
STORE MANAGER
[MAIN HELP]
Net.Commerce Administrator
The tools you need to open a business on the World Wide Web

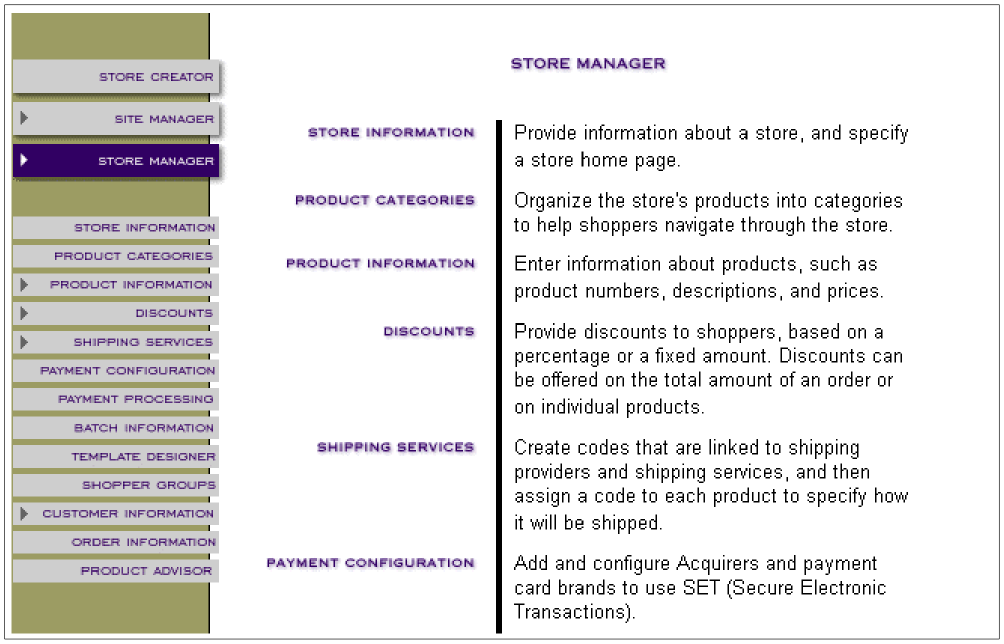
STORE CREATOR
Create a simple store in nine easy steps.

SITE MANAGER
Manage access groups, payment methods, and shipping providers for the site.

STORE MANAGER
Build and manage your interactive catalog, including products, prices, and shipping details.

Figure 329. Net.Commerc Administrator Page

3. Click on **Payment Configuration** from the list to perform the payment configuration (Figure 330).

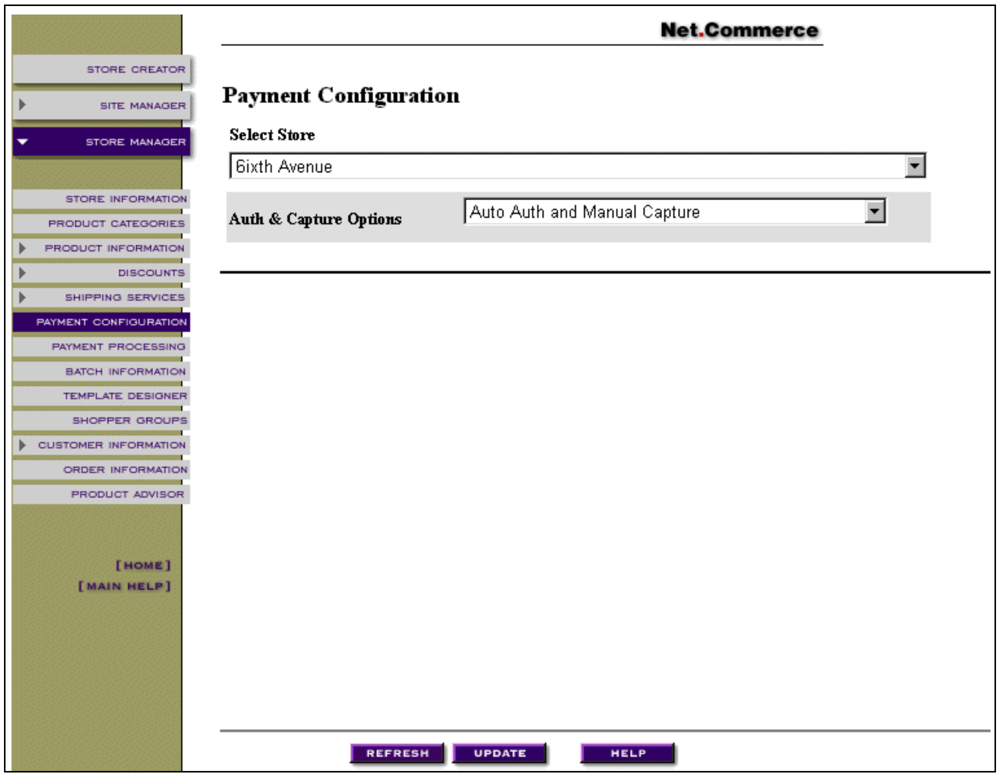


STORE MANAGER

STORE INFORMATION	Provide information about a store, and specify a store home page.
PRODUCT CATEGORIES	Organize the store's products into categories to help shoppers navigate through the store.
PRODUCT INFORMATION	Enter information about products, such as product numbers, descriptions, and prices.
DISCOUNTS	Provide discounts to shoppers, based on a percentage or a fixed amount. Discounts can be offered on the total amount of an order or on individual products.
SHIPPING SERVICES	Create codes that are linked to shipping providers and shipping services, and then assign a code to each product to specify how it will be shipped.
PAYMENT CONFIGURATION	Add and configure Acquirers and payment card brands to use SET (Secure Electronic Transactions).

Figure 330. Net.Commerce Store Manager

4. On the Payment Configuration page (Figure 331 on page 352), select the store where you want to use SET. Select the Authority & Capture Option that you want to use.



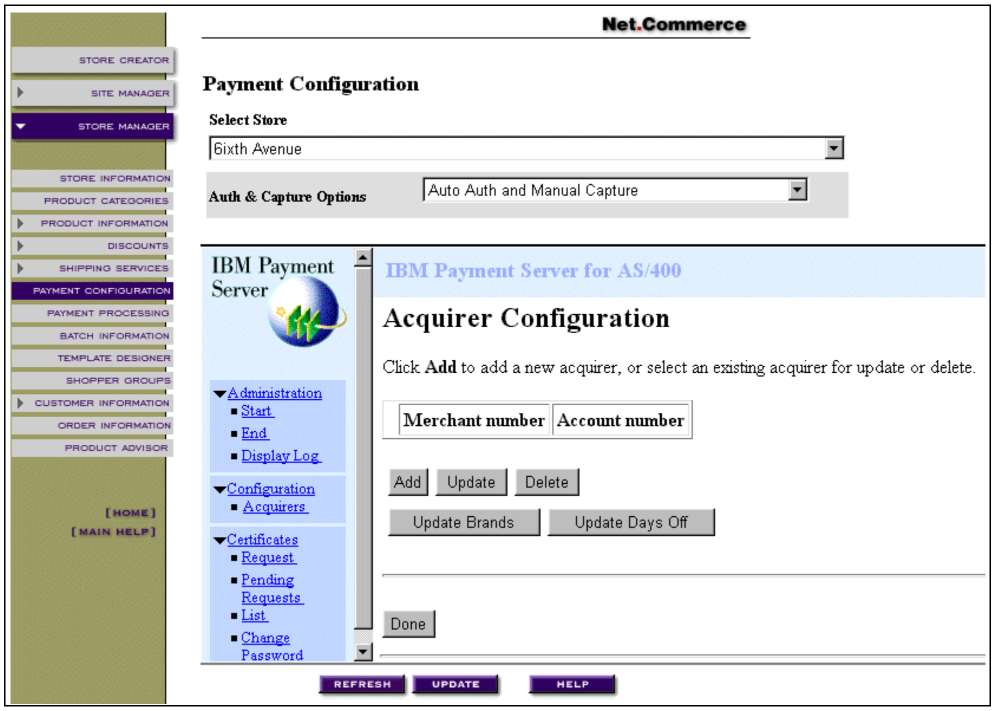
Payment Configuration

Select Store: 6ixth Avenue

Auth & Capture Options: Auto Auth and Manual Capture

Figure 331. Net.Commerce Payment Configuration

5. Click **Update**. This adds the payment system for the selected merchant to the Payment Server. The Net.Commerce Payment Acquirer Configuration page (Figure 332) appears.



Payment Configuration

Select Store: 6ixth Avenue

Auth & Capture Options: Auto Auth and Manual Capture

Acquirer Configuration

Click **Add** to add a new acquirer, or select an existing acquirer for update or delete.

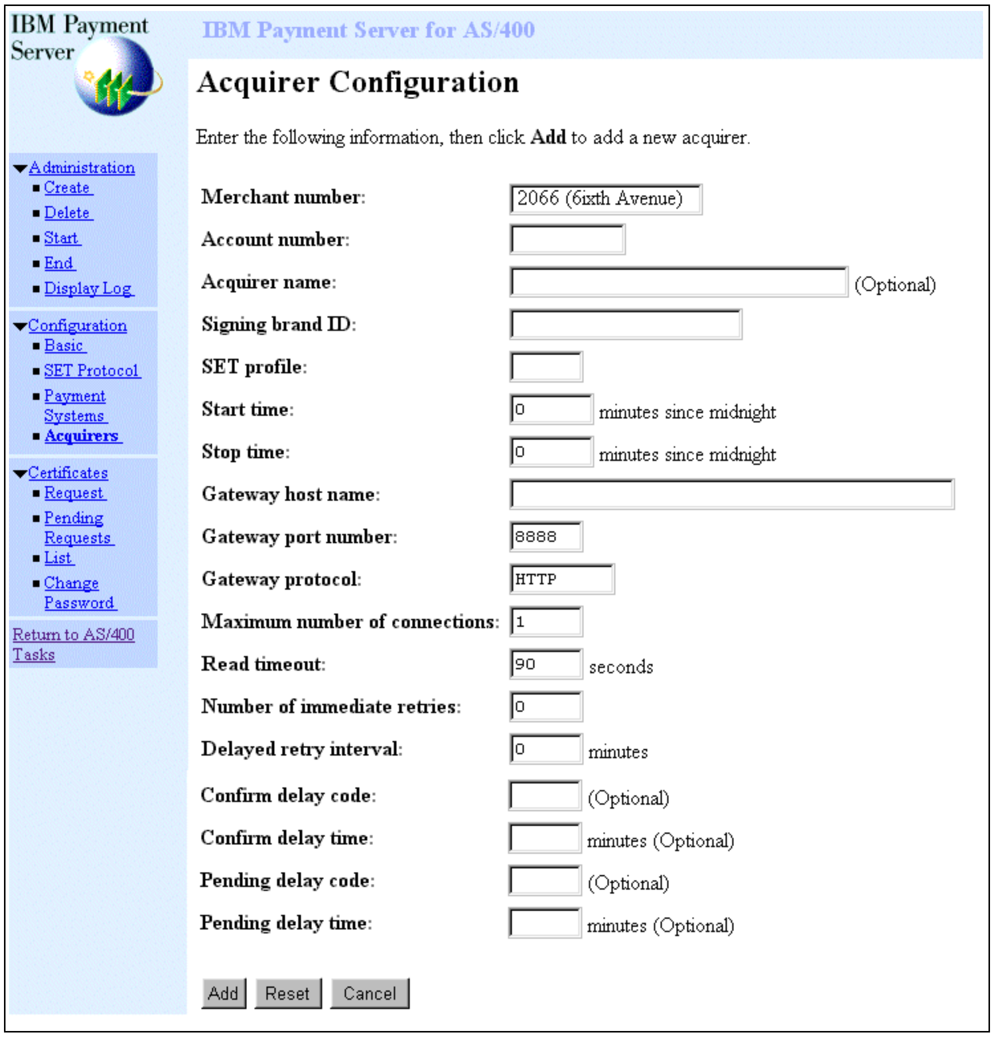
Merchant number	Account number

Figure 332. Net.Commerce Payment Acquirer Configuration

16.1.6 Acquirer Configuration of the Payment Server

The next step in the configuration of the Payment Server is to configure the acquirers. Follow these steps:

1. On the Acquirer Configuration page (Figure 332 on page 352), click **Add**. The Acquirer Configuration Form page (Figure 333) appears.
2. On the Acquirer Configuration Form page (Figure 333), complete the information that you received from your acquirer. The merchant number is completed based on the store that you selected in Figure 331 on page 352. Use your values from Table 7 on page 98 as input to your fields.



Acquirer Configuration

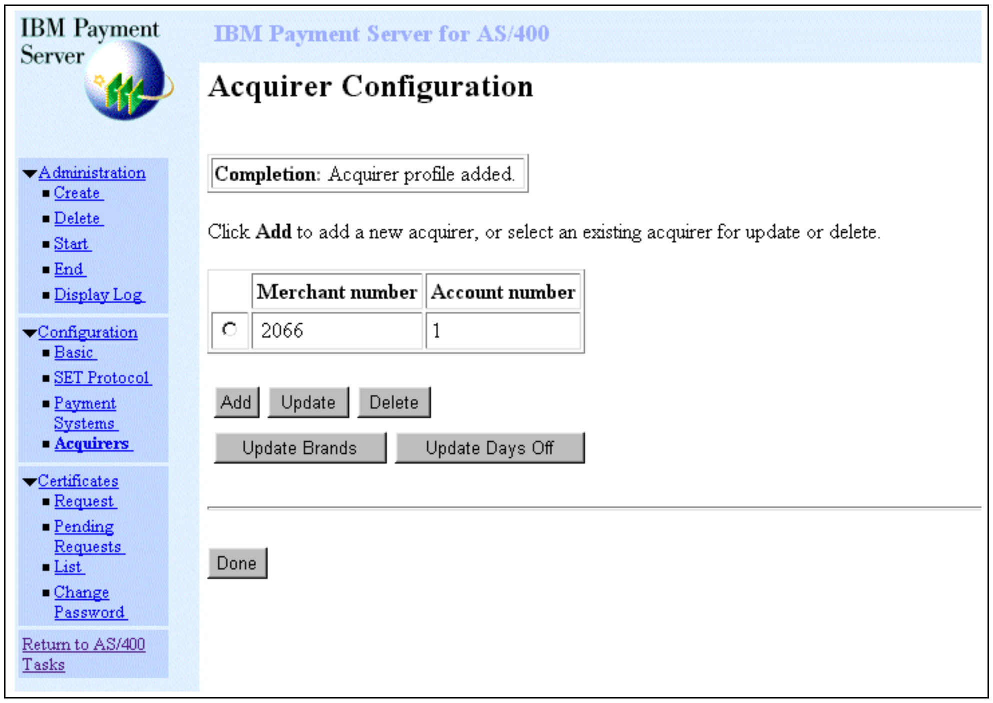
Enter the following information, then click **Add** to add a new acquirer.

Merchant number:	2066 (6ixth Avenue)
Account number:	
Acquirer name:	(Optional)
Signing brand ID:	
SET profile:	
Start time:	0 minutes since midnight
Stop time:	0 minutes since midnight
Gateway host name:	
Gateway port number:	8888
Gateway protocol:	HTTP
Maximum number of connections:	1
Read timeout:	90 seconds
Number of immediate retries:	0
Delayed retry interval:	0 minutes
Confirm delay code:	(Optional)
Confirm delay time:	minutes (Optional)
Pending delay code:	(Optional)
Pending delay time:	minutes (Optional)

Figure 333. Payment Server Acquirer Configuration Form Page

3. Click **Add**.

   You receive a message that the acquirer profile is added (Figure 334 on page 354).



Acquirer Configuration

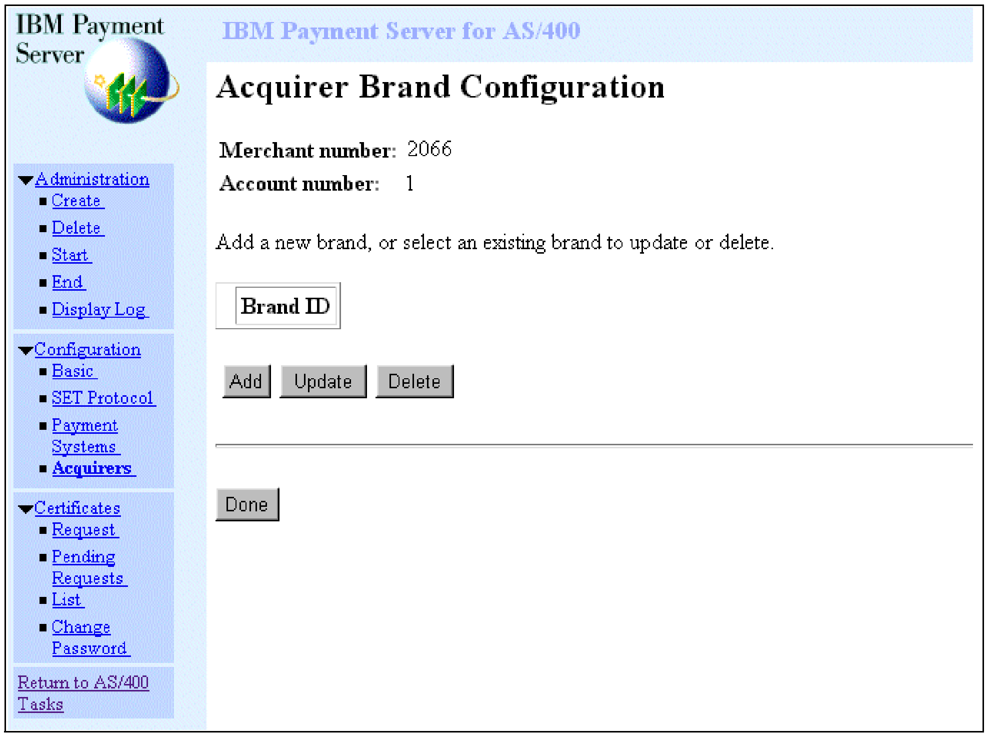
Completion: Acquirer profile added.

Click **Add** to add a new acquirer, or select an existing acquirer for update or delete.

	Merchant number	Account number
○	2066	1

Figure 334. Payment Server Acquirer Profile Added Page

4. Select the merchant, and click **Update**.
5. On the Acquirer Brand Configuration page (Figure 335), click **Add**.



Acquirer Brand Configuration

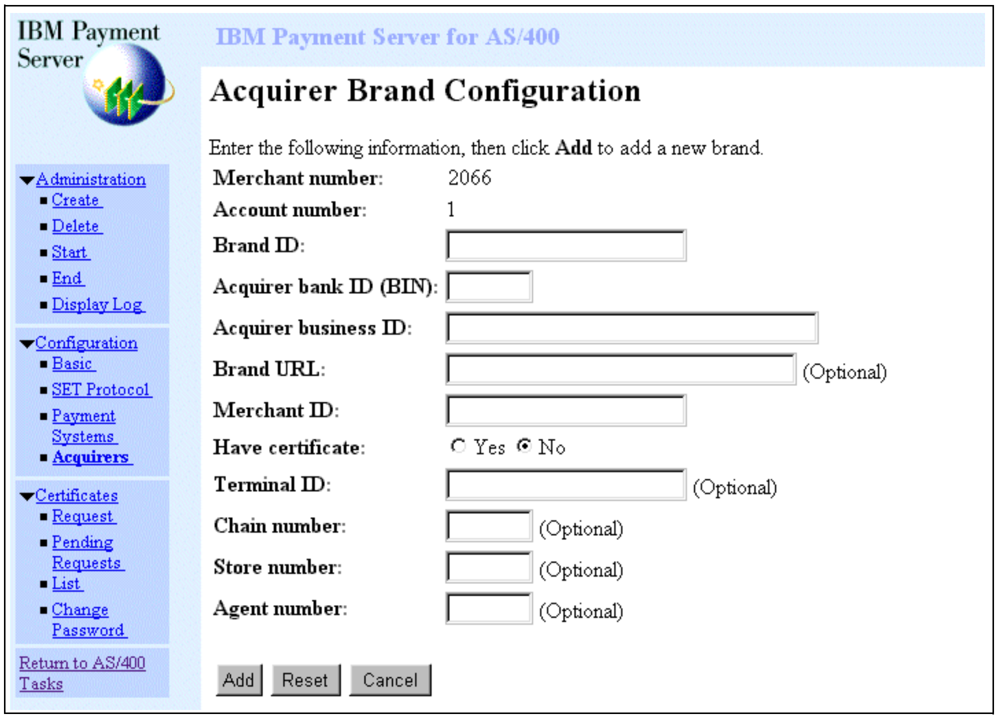
Merchant number: 2066
Account number: 1

Add a new brand, or select an existing brand to update or delete.

Brand ID

Figure 335. Payment Server Acquirer Add Brand Page

6. On the Acquirer Brand Configuration Form page (Figure 336), enter the information you received from your acquirer. Use your values from Table 8 on page 99 as input to your fields.



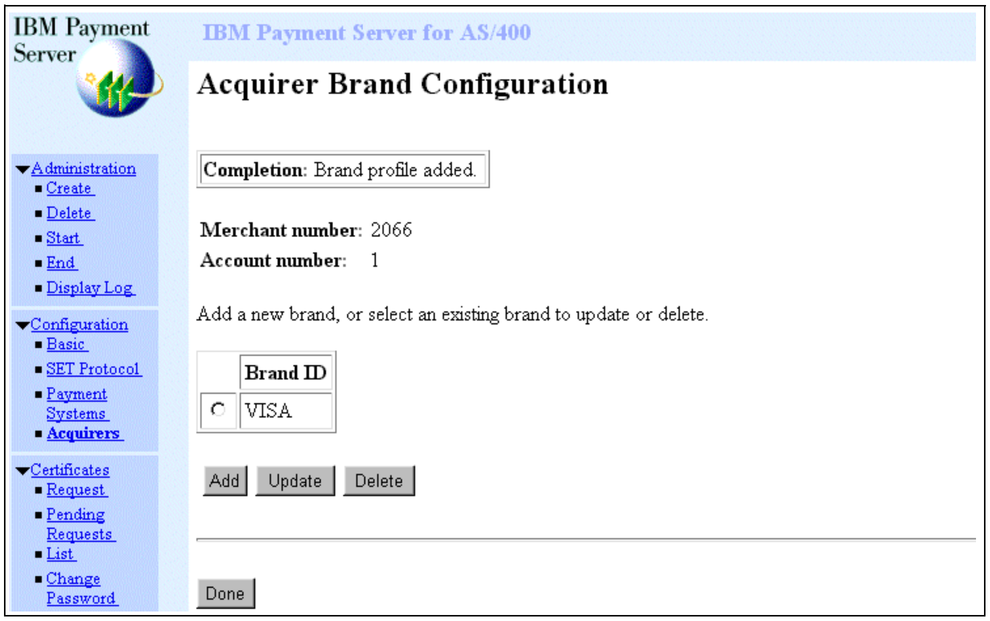
Acquirer Brand Configuration

Enter the following information, then click **Add** to add a new brand.

Merchant number:	2066
Account number:	1
Brand ID:	
Acquirer bank ID (BIN):	
Acquirer business ID:	
Brand URL:	(Optional)
Merchant ID:	
Have certificate:	Yes [✓ No]
Terminal ID:	(Optional)
Chain number:	(Optional)
Store number:	(Optional)
Agent number:	(Optional)

Figure 336. Payment Server Acquirer Brand Configuration Form Page

7. Click **Add**. The Payment Server Acquirer Brand Profile Added page (Figure 337) displays.



Acquirer Brand Configuration

Completion: Brand profile added.

Merchant number: 2066
Account number: 1

Add a new brand, or select an existing brand to update or delete.

	Brand ID
○	VISA

Figure 337. Payment Server Acquirer Brand Profile Added Page

16.1.7 SET Certificate

The digital certificate that is used in SET is not the same digital certificate that is used during SSL. The SET process uses special certificates. It also uses 128-bit encryption for the credit card information, even outside of North America.

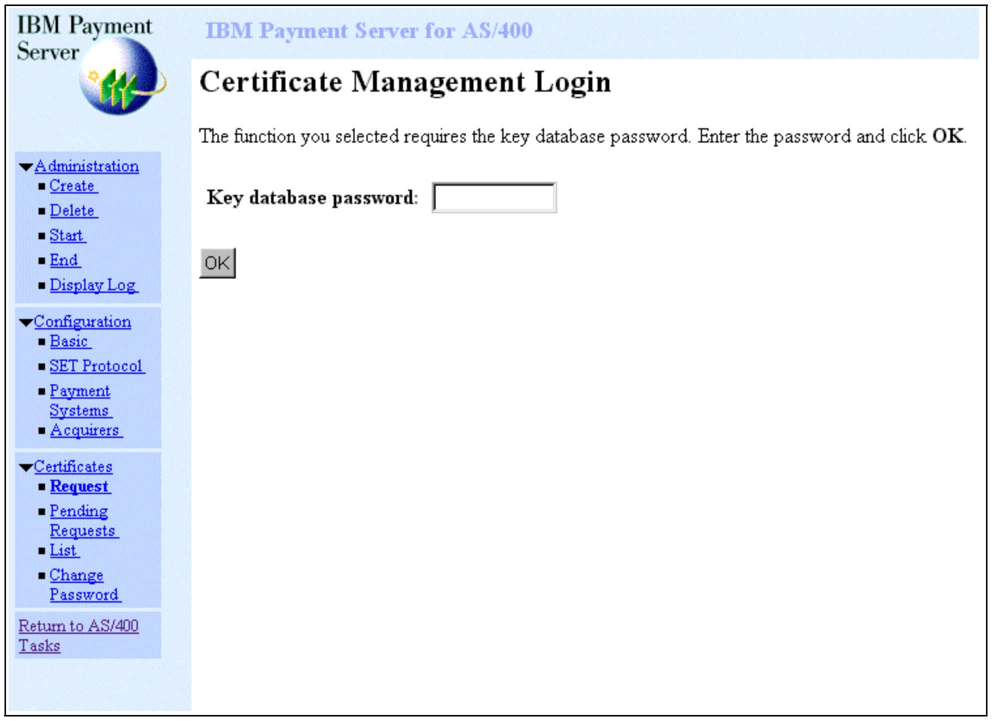
To use SET as a merchant, you must register with a certificate authority (CA) before you can receive SET payment instructions from cardholders or process SET transactions through a payment gateway. You also need a copy of the registration form from your financial institution. Your software must identify the acquirer to the CA.

16.1.8 Requesting a SET Merchant Certificate from a CA

To conduct commercial business on the Internet, you should request your SET merchant certificate from a certificate authority, such as VeriSign or Globeset. This section describes how to obtain a SET merchant certificate from a certificate authority. To use SET for secure payment, your server must have a digital certificate.

When you choose to use a CA to issue a server certificate, you must first request the certificate. Follow these steps:

1. From the IBM Payment Server for AS/400 page, click **Certificates** in the left-hand frame to display an extended list of server tasks.
2. Click **Request** from the list to request a certificate.
3. On the Certificate Management Login Page (Figure 338), enter your Key database password that you selected when you created your Payment Server in 16.1.2, “Creating a Payment Server” on page 344.



Certificate Management Login

The function you selected requires the key database password. Enter the password and click **OK**.

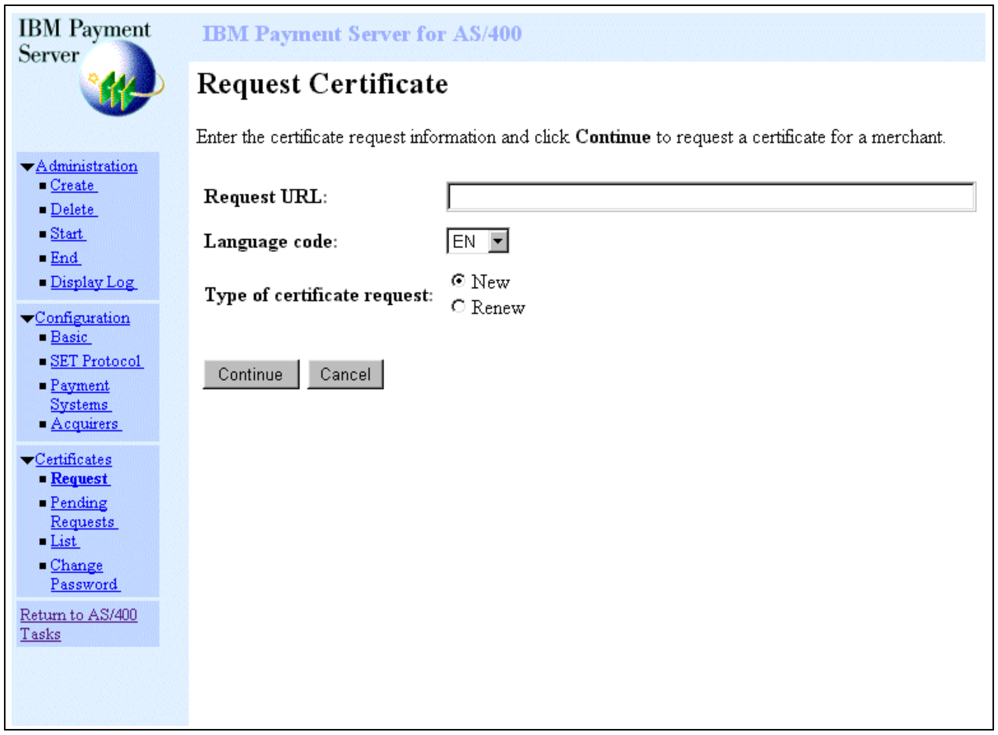
Key database password:

OK

Figure 338. Payment Server Certificate Management Login Page

Click **OK** to display the Request Certificate page.

4. Enter the URL to your CA in the Request URL field. Click **Continue**. The Payment Server Certificate Request page (Figure 339) is displayed.



Request Certificate

Enter the certificate request information and click **Continue** to request a certificate for a merchant.

Request URL:

Language code: EN

Type of certificate request: [✓ New] Renew

Figure 339. Payment Server Certificate Request

Click **Continue**.

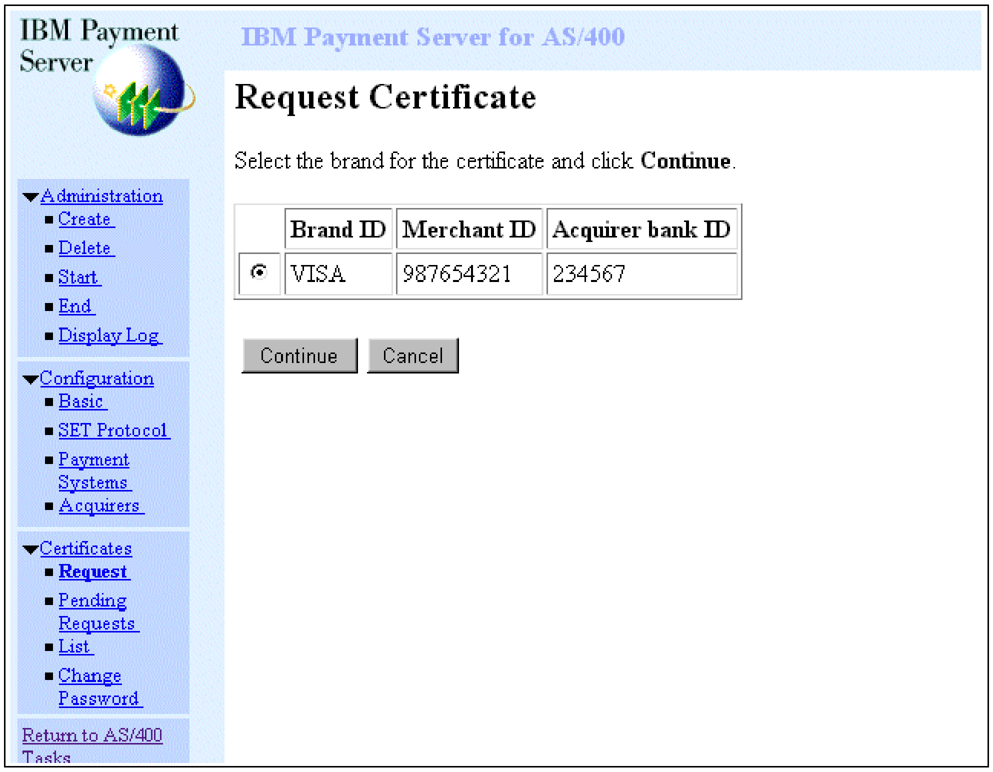
If you do not already have a valid SET certificate installed, you will be prompted to enter a root hash code that you received from your acquirer.

Click **Continue** to display the Payment Server Request Certificate Brand page.

5. On the Payment Server Request Certificate Brand page, select the brand for the certificate and click **Continue** (Figure 340 on page 358).

Important

It is important that you configure acquirers and brands first with the information received from the acquirer.



Request Certificate

Select the brand for the certificate and click **Continue**.

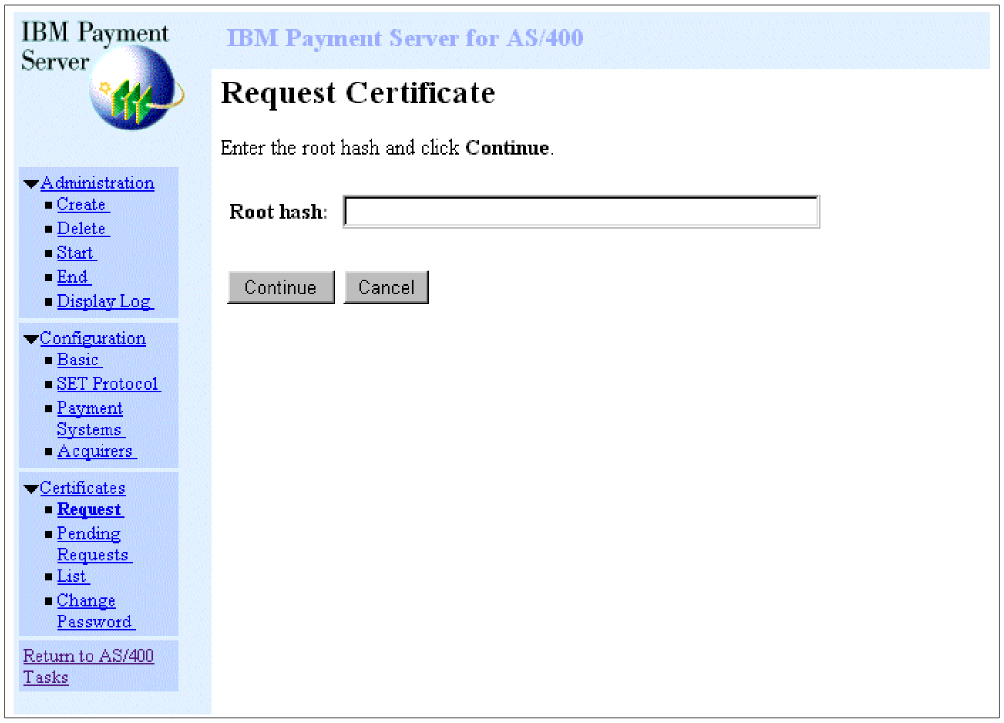
	Brand ID	Merchant ID	Acquirer bank ID
●	VISA	987654321	234567

Figure 340. Payment Server Request Certificate Brand

6. You may be asked to enter the root hash, which is a 40-character string.

   On the Payment Server Certificate Root Hash page, enter the root hash code that you received from your acquirer (Figure 341 on page 359).

   Click **Continue**.



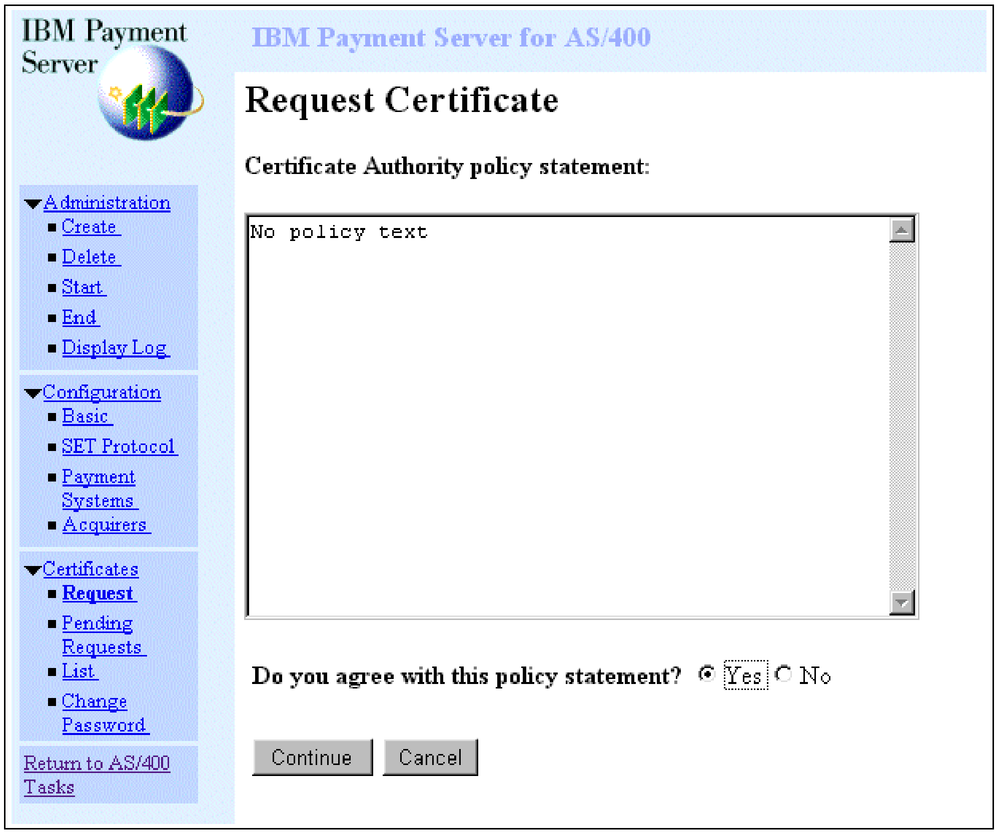
Request Certificate

Enter the root hash and click **Continue**.

Root hash:

Figure 341. Payment Server Certificate Root Hash

7. On the Payment Server Certificate Request Policy page, you have to read and agree with the policy (Figure 342 on page 360). Click **Continue**.



Request Certificate

Certificate Authority policy statement:

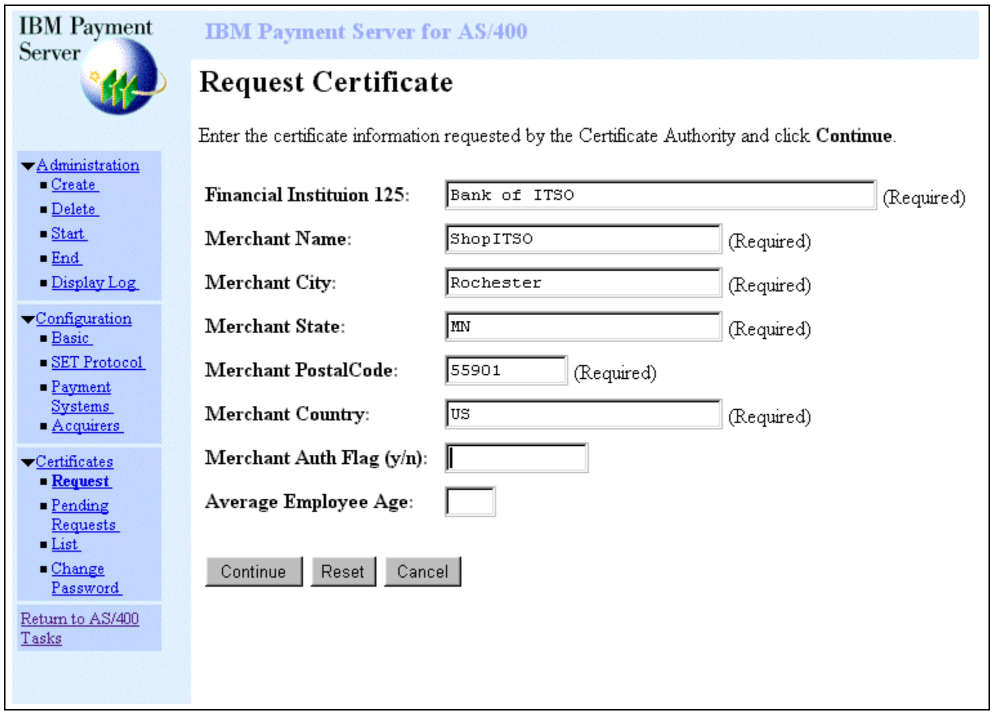
```
No policy text
```

Do you agree with this policy statement? [✓ Yes] No

Figure 342. Payment Server Certificate Request Policy

8. On the Payment Server Certificate Request Information page (Figure 343 on page 361), enter the certificate information requested by the certificate authority. Use your values from Table 9 on page 100 as input to your fields.

   Click **Continue**.



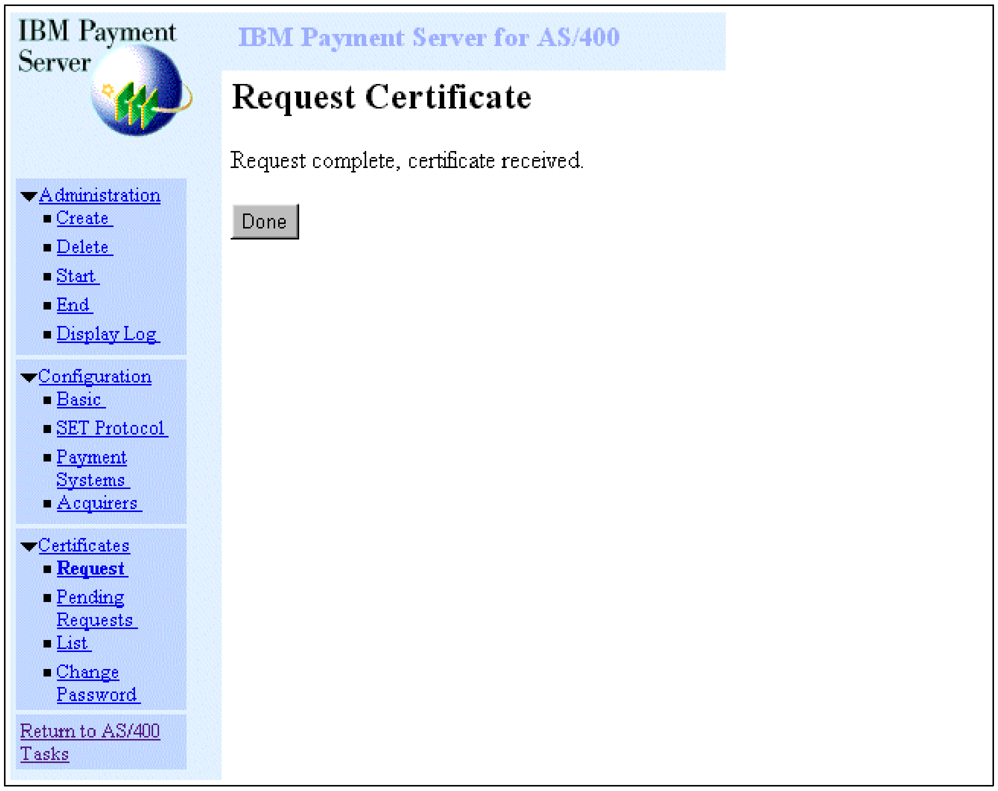
Request Certificate

Enter the certificate information requested by the Certificate Authority and click **Continue**.

Financial Instituion 125:	Bank of ITSO	(Required)
Merchant Name:	ShopITSO	(Required)
Merchant City:	Rochester	(Required)
Merchant State:	MN	(Required)
Merchant PostalCode:	55901	(Required)
Merchant Country:	US	(Required)
Merchant Auth Flag (y/n):		
Average Employee Age:		

Figure 343. Payment Server Certification Request Information Page

9. The Payment Server Certificate Request Complete page is displayed (Figure 344), and the certification receive process is finished. Click **Done**.



Request Certificate

Request complete, certificate received.

Figure 344. Payment Server Certificate Request Complete Page

You have now installed your brand's digital SET certificate on your Payment Server.

16.1.9 Starting and Ending the Payment Server

The Payment Server checks the date, time, and time zone when you use it for shopping. It is important that the system values for the date, time, and time zone in your AS/400 system is correctly set. The Payment Server also checks the date, time, and time zone that is sent to it from the eWallet, and compares it with its own values.

To display your date-and-time-related system values, complete these steps:

1. On an AS/400 command line, type:

   ```
   DSPSYSVAL SYSVAL(QDATE)
   ```

   Press **Enter** to see the Date System Value.

   If the value is incorrect, you have to change it. On an AS/400 command line, type:

   ```
   CHGSYSVAL SYSVAL(QDATE) VALUE('mm/dd/yy')
   ```

   In this statement, `mm/dd/yy` occurs in the command, where `mm` = month, `dd` = day, and `yy` = year.

   Note

   The format of the date field is not the same in all countries. Verify your format before you change it.

2. On an AS/400 command line, type:

   ```
   DSPSYSVAL SYSVAL(QTIME)
   ```

   Press **Enter** to see the Time System Value. If the value is incorrect, you have to change it. On an AS/400 command line, type:

   ```
   CHGSYSVAL SYSVAL(QTIME) VALUE('hh:mm:ss')
   ```

   In this statement, `hh` = hour, `mm` = minute, and `ss` = second.

3. On an AS/400 command line, type:

   ```
   DSPSYSVAL SYSVAL(QUTCOFFSET)
   ```

   Press **Enter** to see the Coordinated Universal Time Offset System Value. This value specifies the difference in hours and minutes between UTC, also known as Greenwich mean time (GMT), and the current system time.

   If the value is incorrect, you must change it. On an AS/400 command line, type:

   ```
   CHGSYSVAL SYSVAL(QUTCOFFSET) VALUE('-hh:mm')
   ```

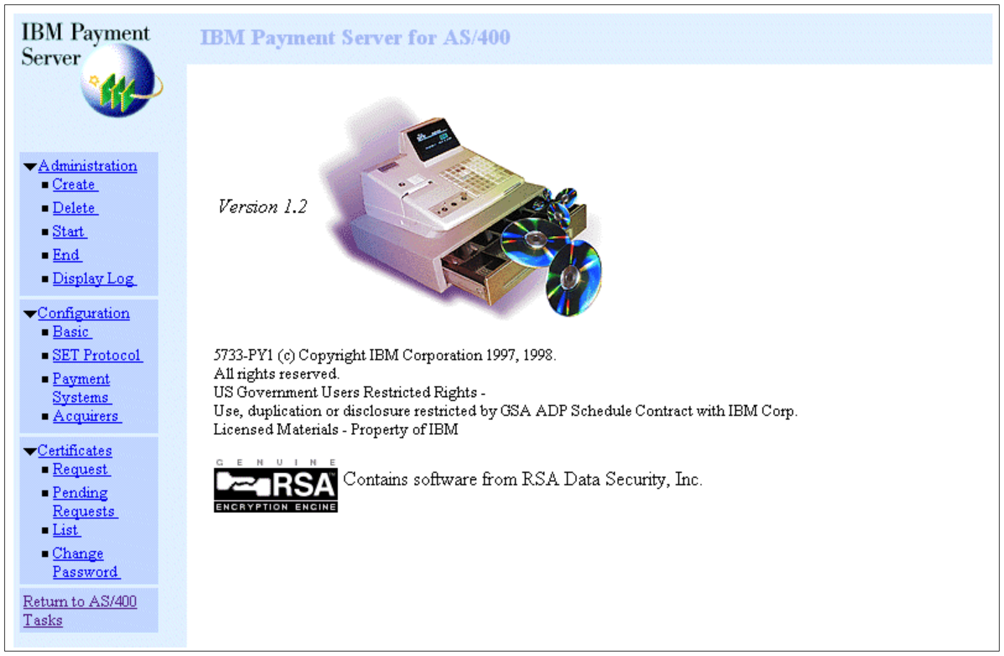
   In this statement, the symbol "-" can be either "+" or "-" depending on where your time zone is in relation with GMT. `hh` = hour and `mm` = minute.

16.1.9.1 Starting the Payment Server

The Payment Server starts automatically when you start your Net.Commerce instance. There are two ways to start your Payment Server manually.

One way is to start the Payment Server from the IBM Payment Server for AS/400 page (Figure 345). To do so, follow these steps:

1. Click **Administration —> Start**.

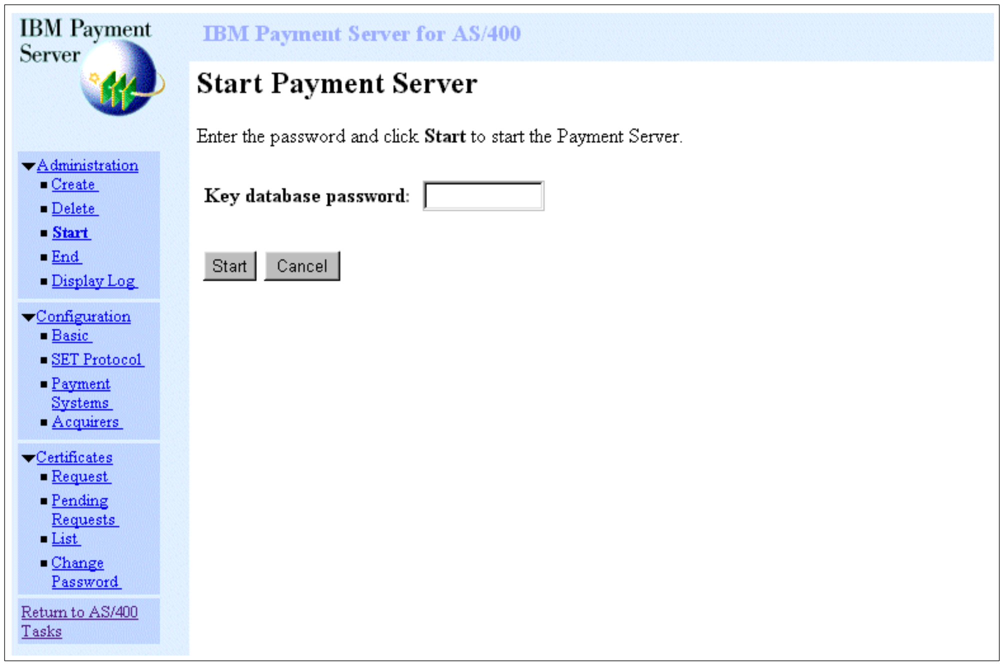


5733-PY1 (c) Copyright IBM Corporation 1997, 1998.
All rights reserved.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Licensed Materials - Property of IBM

Contains software from RSA Data Security, Inc.

Figure 345. Payment Server Page

2. On the **Start Payment Server** page (Figure 346), enter the Key database password. Click **Start**.



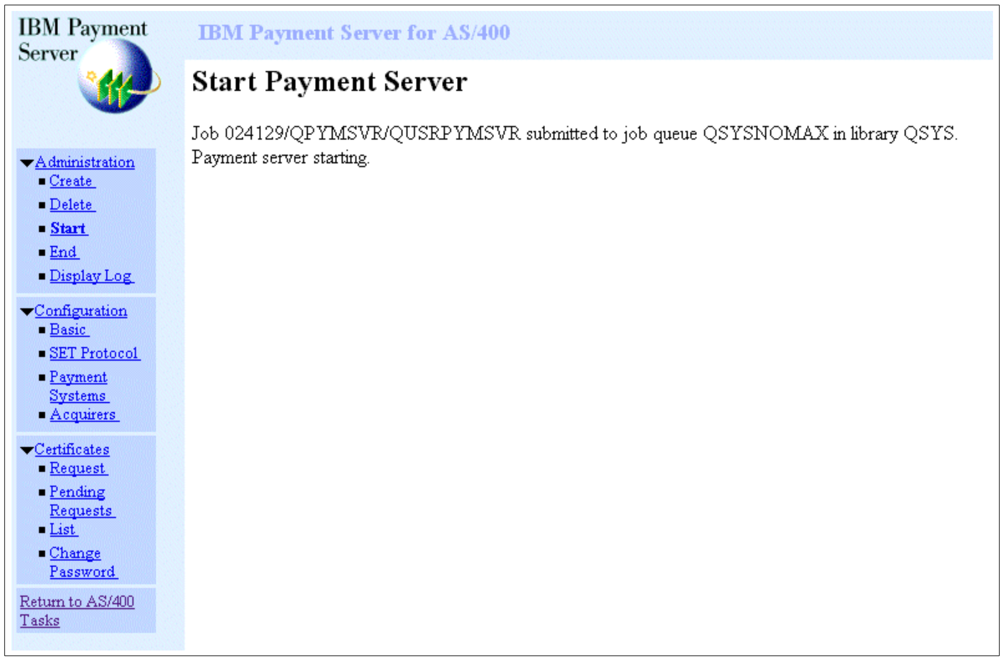
Start Payment Server

Enter the password and click **Start** to start the Payment Server.

Key database password:

Figure 346. Start Payment Server

Then, the Payment Server starts as shown in Figure 347 on page 364.



Job 024129/QPYMSVR/QUSRPYMSVR submitted to job queue QSYSNOMAX in library QSYS.
Payment server starting.

Figure 347. Payment Server Starting Page

The other way to start your Payment Server is to use AS/400 commands:

1. On an AS/400 command line, type:

   ```
   STRPYMSVR KEYPWD(password)
   ```

   Where `password` occurs in the command, type the Key database password.

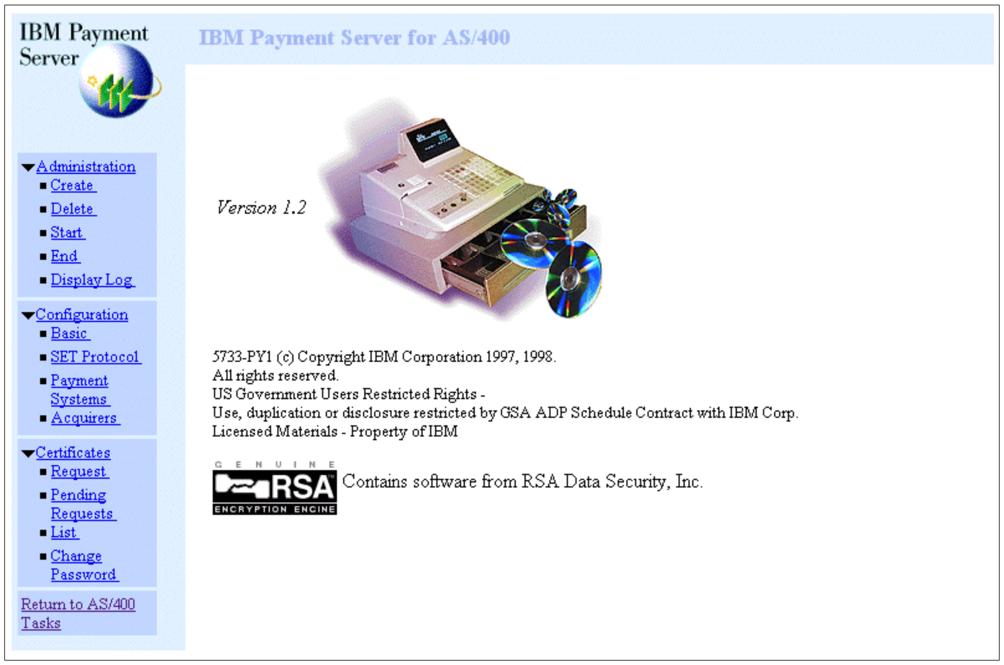
2. Press **Enter**. The following message appears: `Payment server starting`.

The Payment Server job QUSRPYMSVR is now running in subsystem QSYSWRK.

16.1.9.2 Ending the Payment Server

There are two ways to end your Payment Server. The first way is to end it from the IBM Payment Server for AS/400 page (Figure 348 on page 365). The steps for this process are described here:

1. Click **Administration —> End**.

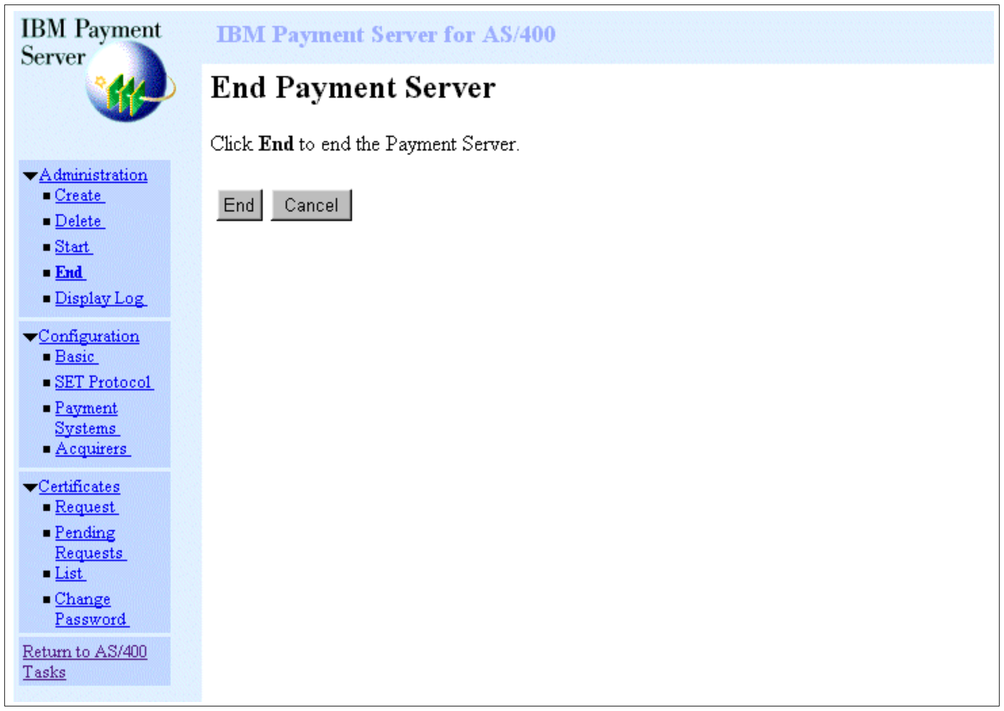


5733-PY1 (c) Copyright IBM Corporation 1997, 1998.
All rights reserved.
US Government Users Restricted Rights -
Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Licensed Materials - Property of IBM

Contains software from RSA Data Security, Inc.

Figure 348. Payment Server Page

2. On the End Payment Server page (Figure 349), enter the Key database password. Click **End**.

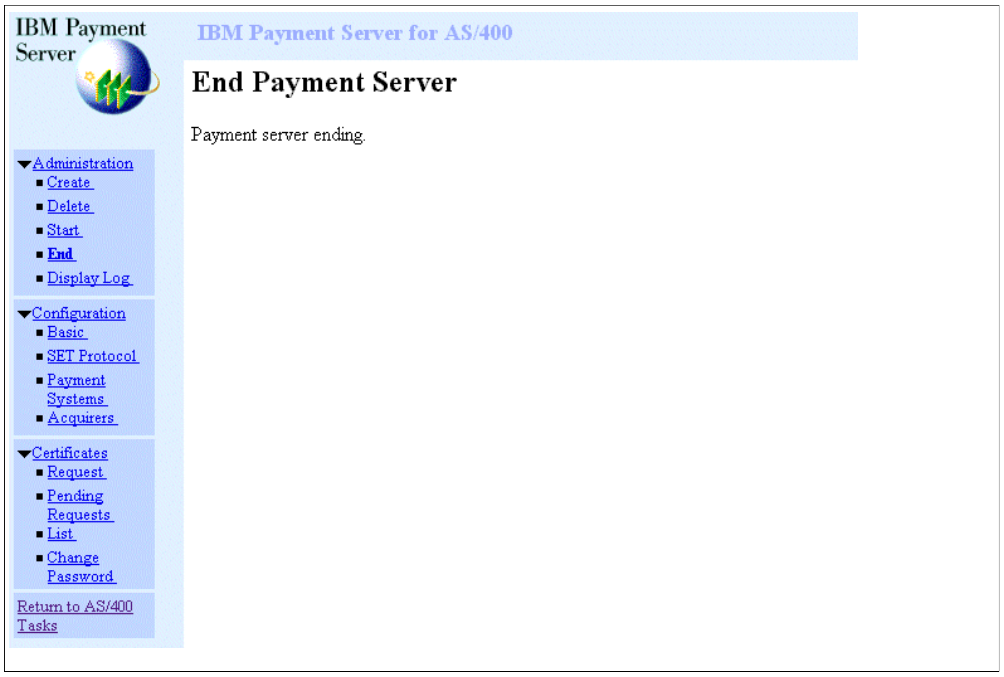


End Payment Server

Click **End** to end the Payment Server.

Figure 349. End Payment Server

The Payment Server ends, as shown in Figure 350 on page 366.



End Payment Server

Payment server ending.

Figure 350. Payment Server Ending

The other way to end your Payment Server is to use AS/400 commands. This process is described here:

1. On an AS/400 command line, type:

```
ENDSTRPYMSVR
```

2. Press **Enter**. The following message appears: `Payment Server ending`.

The Payment Server job QUSRPYMSVR is now ending in the subsystem QSYSWRK.

16.2 Payment Server Payment Processing

To enable manual capture, authorization and credit transactions, and reversals of payments, you can use the Store Manager. The Store Manager lets you mark the orders that you wish to process.

Capture is the process by which your acquirer receives the payment from the customer’s financial institution and remits the payment to you. Orders that are awaiting capture are in the "Capture Ready" state. You can use the Store Manager to change their state to "Capture Requested." The Net.Commerce Background server wakes up periodically, finds the orders that are marked for processing, sends the requests to the acquirer, waits for the responses, and changes the states according to the responses.

16.2.1 Managing Payment Transactions

You can use the Store Manager to perform the following payment functions:

- Search the payment transactions on the database
- Request authorization on a payment transaction

- Request authorization reversal on a payment transaction
- Request capture on a payment transaction
- Request capture reversal on a payment transaction
- Request credit on a payment transaction
- Request credit reversal on a payment transaction
- Reset to the previous manual state

16.2.2 Types of Payment Server Functions

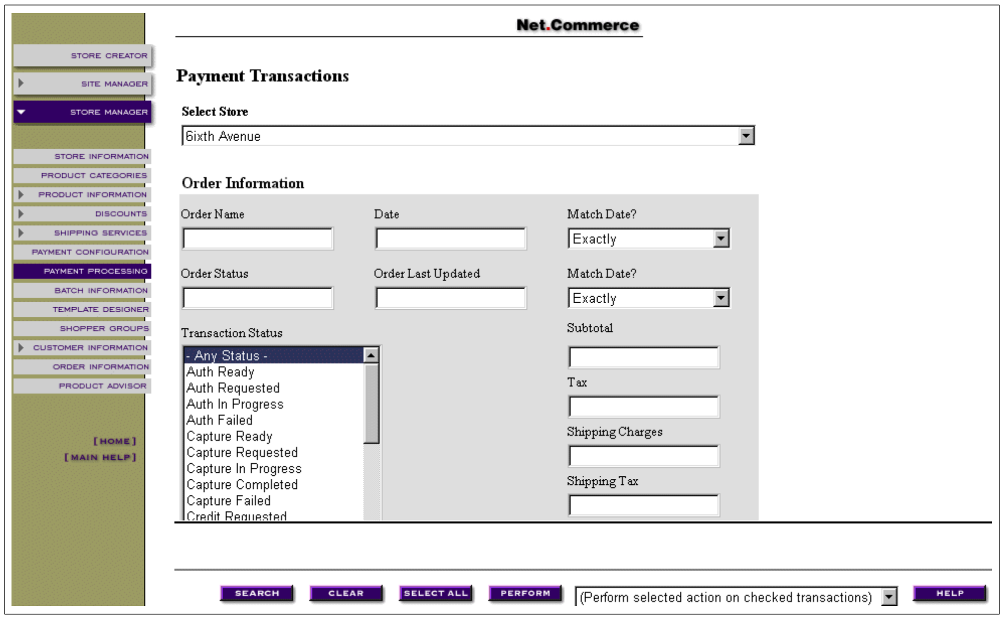
You can use the Store Manager to process the following transactions manually:

- **Request Authorization** — You wish to receive authorization for the customer's purchase to fulfill the order. The order must be in the *Auth Ready* state.
- **Request Authorization Reversal** — The purchase is authorized by the cardholder's financial institution, but you have not yet requested capture. Plus, you wish to cancel the purchase or change the approved amount (downward). The order must be in the *Capture Ready* state.
- **Request Capture** — The transaction is authorized by the customer's financial institution, and you wish to receive payment from your acquirer. The order must be in the *Capture Ready* state.
- **Request Capture Reversal** — The capture has succeeded, but you have not yet received payment. Now, you wish to reverse all or part of the capture. The order must be in a *Capture Completed* state.
- **Request Credit** — Payment has already been made, but you wish to give the customer a credit for all or part of the purchase. The order must be in a *Capture Completed* or *Credit Completed* state (the latter because you can request credit on an order more than once).
- **Request Credit Reversal** — Credit to the customer succeeded, but the customer has not yet received a refund. Plus, you wish to cancel the credit. For example, perhaps the customer decided to keep the goods after all, or the initial credit request was an error. The order must be in a *Credit Completed* state.

16.2.3 Searching the Payment Transactions in the Database

To find the Payment transactions you wish to process, follow these steps:

1. Open Net.Commerce Administrator.
2. On the task bar, click **Store Manager**. Then, select **Payment Processing** (Figure 351 on page 368).



Payment Transactions

Select Store
6ixth Avenue

Order Information

Order Name | Date | Match Date? Exactly

Order Status | Order Last Updated | Match Date? Exactly

Transaction Status: - Any Status -, Auth Ready, Auth Requested, Auth In Progress, Auth Failed, Capture Ready, Capture Requested, Capture In Progress, Capture Completed, Capture Failed, Credit Requested

Subtotal

Tax

Shipping Charges

Shipping Tax

(Perform selected action on checked transactions)

Figure 351. Payment Transaction Database Page

You can search the database for information contained in any of the non-italicized fields on the form that appears.

3. Click **Search**.

A list of payment transactions in the database appears in the bottom frame.

16.2.4 Requesting Authorization on a Payment Transaction

After a customer decides to make a purchase, the first step in the payment cycle is to have the transaction authorized. To do this, the Payment Server must send the purchase information to an acquirer’s Payment Gateway. The acquirer communicates with the customer’s financial institution over legacy networks. If the purchase is acceptable to the customer’s institution, the acquirer sends you an authorization message. If the purchase is not acceptable, a failed authorization message is sent.

Normally, you wait until you receive authorization, fulfill the order, and then initiate the capture. However, it is possible to handle these steps automatically (for example, if you are selling information or software that can be fulfilled directly from the store). Whether the system manages capture automatically or manually is determined during the acquirer configuration.

If you set up your Net.Commerce server for manual authorization, you must manually request authorization. The authorization request is done from Payment Processing, under Store Manager. For further information, refer to *Net.Commerce for AS/400: The Net.Commerce Payment Module Version 3.2*. You can find it in the AS/400 file system /Qibm/ProdData/NetCommerce/html/Mri2924/ncbooks/set.pdf.

16.2.5 Requesting Authorization Reversal on a Payment Transaction

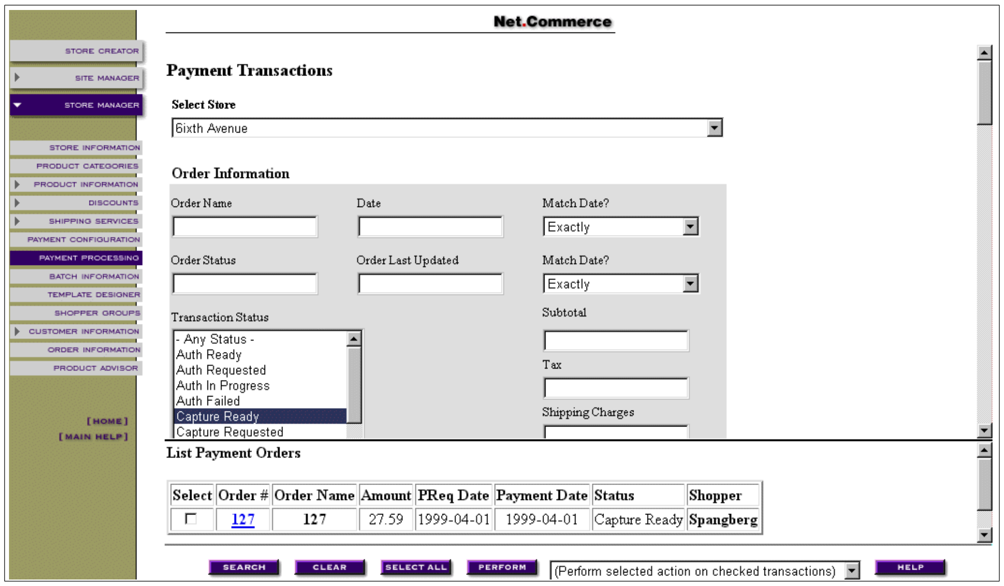
After a transaction is authorized, and before capture is initiated, it is possible to reverse the authorization.

The authorization reversal request is done from the Payment Processing under Store Manager. For further information, refer to *Net.Commerce for AS/400: The Net.Commerce Payment Module Version 3.2*. You can find this document in the AS/400 file system /Qibm/ProdData/NetCommerce/html/Mri2924/ncbooks/set.pdf.

16.2.6 Requesting Capture on a Payment Transaction

Once the transaction is authorized and the order is fulfilled, you can request capture. Capture is the process by which your acquirer receives the payment from the customer's financial institution and remits the payment to you. To request capture, complete the following steps:

1. Open Net.Commerce Administrator.
2. On the task bar, click **Store Manager**, and then click on **Payment Processing**.
3. From the **Select Store** drop-down list, select the appropriate store.
4. From the SET Transaction Status window, select **Capture Ready**.
5. Click **Search**. The information about transactions whose status is "Capture Ready" appears in the bottom frame (Figure 352).



Payment Transactions

Select Store: 6ixth Avenue

Order Information

Order Name:	Date:	Match Date? Exactly
Order Status:	Order Last Updated:	Match Date? Exactly
Transaction Status:		Subtotal:
		Tax:
		Shipping Charges:

Transaction Status list: - Any Status -, Auth Ready, Auth Requested, Auth In Progress, Auth Failed, Capture Ready (selected), Capture Requested

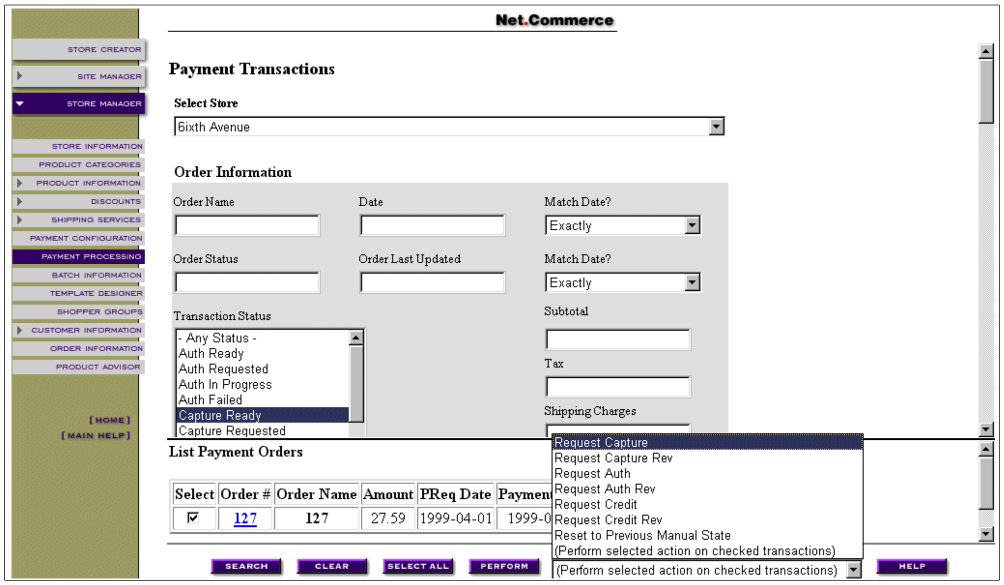
List Payment Orders

Select	Order #	Order Name	Amount	PReq Date	Payment Date	Status	Shopper
☐	127	127	27.59	1999-04-01	1999-04-01	Capture Ready	Spangberg

(Perform selected action on checked transactions)

Figure 352. Payment Processing — Capture Ready

6. Select the order or orders for which you wish to request capture by clicking their **Select** check boxes. To process all the orders, click the **Select All** button.
7. Click **Perform selected action on checked transactions** to open the pop-up window. Select **Request Capture** (Figure 353 on page 370).



Payment Transactions

Select Store: 6ixth Avenue

Order Information

Order Name | Date | Match Date? Exactly

Order Status | Order Last Updated | Match Date? Exactly

Transaction Status: - Any Status -, Auth Ready, Auth Requested, Auth In Progress, Auth Failed, Capture Ready, Capture Requested

Subtotal

Tax

Shipping Charges

List Payment Orders

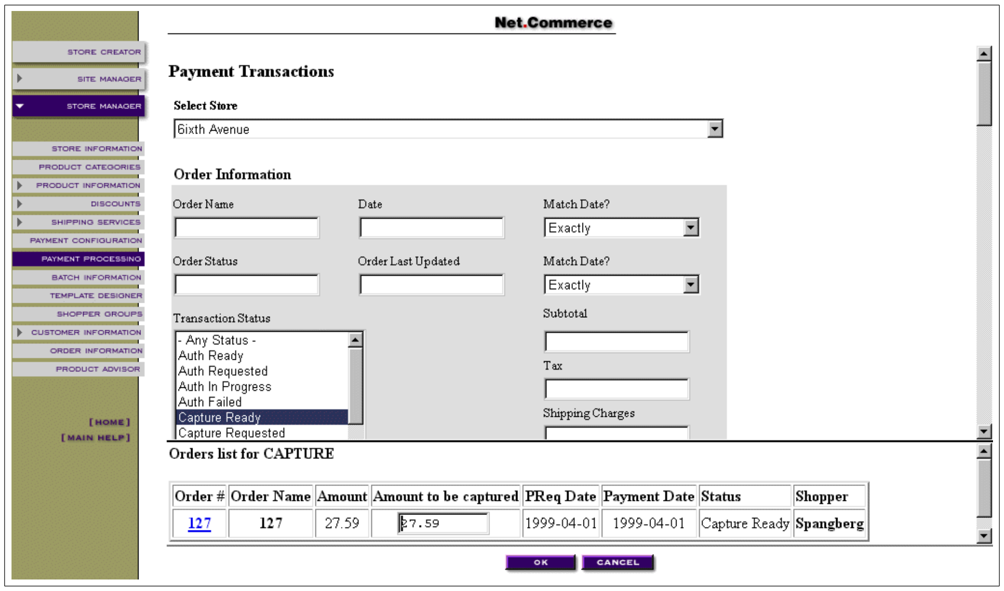
Select	Order #	Order Name	Amount	PReq Date	Paymen
☑	127	127	27.59	1999-04-01	1999-0

Request Capture
Request Capture Rev
Request Auth
Request Auth Rev
Request Credit
Request Credit Rev
Reset to Previous Manual State
(Perform selected action on checked transactions)

(Perform selected action on checked transactions)

Figure 353. Payment Processing — Request Capture

8. Click **Perform**. A new table appears that shows additional details about the selected orders (Figure 354).



Payment Transactions

Select Store: 6ixth Avenue

Order Information

Order Name | Date | Match Date? Exactly

Order Status | Order Last Updated | Match Date? Exactly

Transaction Status: - Any Status -, Auth Ready, Auth Requested, Auth In Progress, Auth Failed, Capture Ready, Capture Requested

Subtotal

Tax

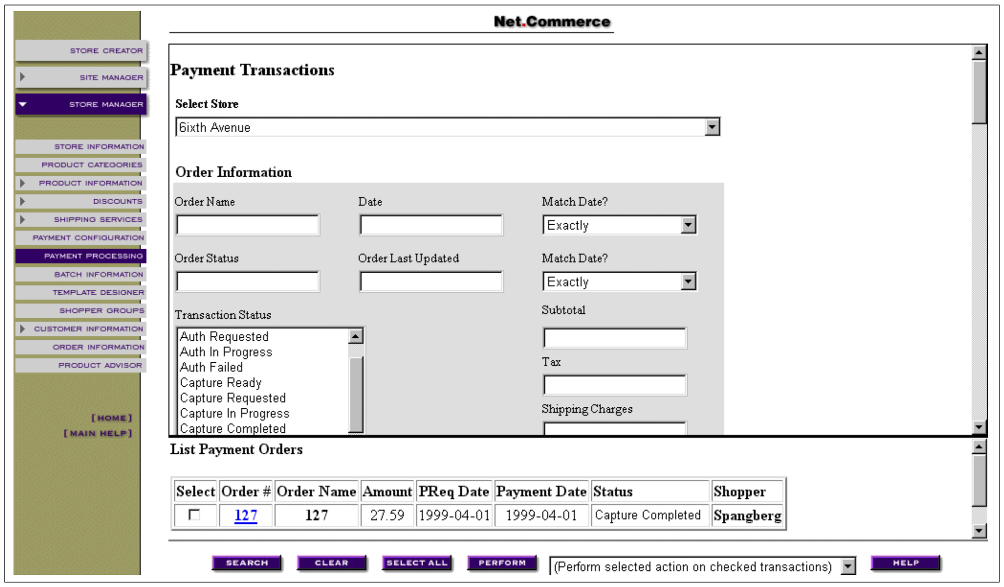
Shipping Charges

Orders list for CAPTURE

Order #	Order Name	Amount	Amount to be captured	PReq Date	Payment Date	Status	Shopper
127	127	27.59	27.59	1999-04-01	1999-04-01	Capture Ready	Spangberg

Figure 354. Payment Processing — Request Capture Details

9. After checking the **Amount to be Captured** column to ensure that the order amounts are accurate, and changing them if necessary, click **OK**.



Payment Transactions

Select Store: 6ixth Avenue

Order Information

Order Name | Date | Match Date? Exactly

Order Status | Order Last Updated | Match Date? Exactly

Transaction Status: Auth Requested, Auth In Progress, Auth Failed, Capture Ready, Capture Requested, Capture In Progress, Capture Completed

Subtotal

Tax

Shipping Charges

List Payment Orders

Select	Order #	Order Name	Amount	PReq Date	Payment Date	Status	Shopper
☐	127	127	27.59	1999-04-01	1999-04-01	Capture Completed	Spangberg

(Perform selected action on checked transactions)

Figure 355. Payment Processing — Amount to Be Captured

A confirmation message appears. The transaction status changes to "Capture Requested."

16.2.7 Requesting Capture upon Order Fulfillment

If you are going to use a back-end system to perform order fulfillment, you may want to request a capture from the acquirer automatically when the order is fulfilled. To do that, you must update the SETSTATUS Net.Commerce table. Refer to 17.4, “Requesting Capture upon Order Fulfillment” on page 398.

16.2.8 Requesting Capture Reversal on a Payment Transaction

When you request a capture, the Payment Server automatically initiates the capture process with the Acquirer’s Payment Gateway. If you need to cancel the capture after the success message is sent from the Payment Gateway but before the payment is received, you can reverse the capture.

You can only request a capture reversal of a completed capture before the batch containing the capture transaction is closed. This is true for both implicit and explicit batch processing. If you are using implicit batch processing, the Payment Gateway rejects the request. If you are using explicit batch processing, the Payment Server rejects the request.

The capture reversal request is done from the Payment Processing under Store Manager. For further information, refer to *Net.Commerce for AS/400: The Net.Commerce Payment Module Version 3.2*. You can find this document in the AS/400 file system /Qibm/ProdData/NetCommerce/html/Mri2924/ncbooks/set.pdf.

16.2.9 Requesting Credit on a Payment Transaction

You can only request a credit on a completed capture after the batch containing the capture transactions closed. This is true for both implicit and explicit batch processing.

The credit request is done from Payment Processing under Store Manager. For further information, refer to *Net.Commerce for AS/400: The Net.Commerce Payment Module Version 3.2*. You can find this document in the AS/400 file system /Qibm/ProdData/NetCommerce/html/Mri2924/ncbooks/set.pdf.

16.2.10 Requesting Credit Reversal on a Payment Transaction

If you requested credit for a customer in error, you can request a *credit reversal* provided that the customer has not already received the credit. You can only request the credit reversal of a completed credit before the batch containing the Credit Transactions closes. This is true for both implicit and explicit batch processing. If you are using implicit batch processing, the Payment Gateway rejects the request. If you are using explicit batch processing, the Payment Server rejects the request.

The credit request is done from Payment Processing under Store Manager. For further information, refer to *Net.Commerce for AS/400: The Net.Commerce Payment Module Version 3.2*. You can find it in the AS/400 file system /Qibm/ProdData/NetCommerce/html/Mri2924/ncbooks/set.pdf.

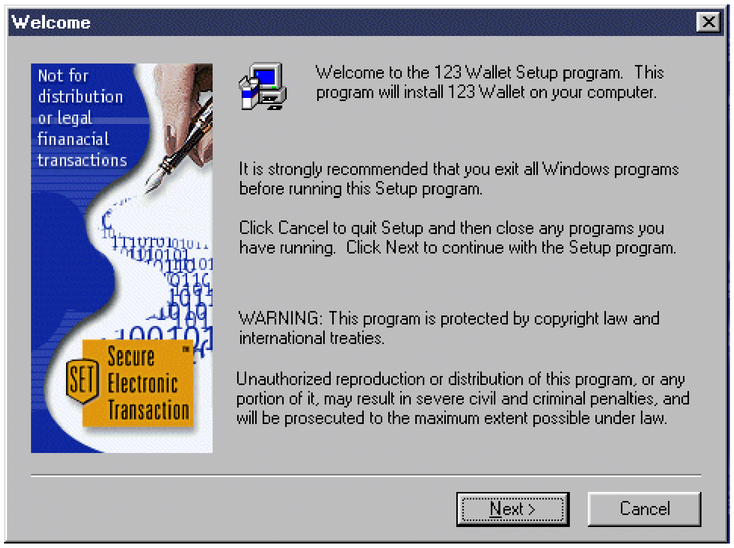
16.3 Installing a SET Compliant eWallet

To show the installation process of a SET compliant eWallet, we are going to use a test version of IBM Consumer Wallet. Complete the following series of tasks:

1. Start the setup program. In our case, the file is:

```
c:\download\ibmwallet981215rel.exe
```

2. From the Welcome window shown in Figure 356, click the **Next** button to begin the installation.


Welcome
Not for distribution or legal finanacial transactions
SET Secure Electronic Transaction
Next >
Cancel

Welcome to the 123 Wallet Setup program. This program will install 123 Wallet on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

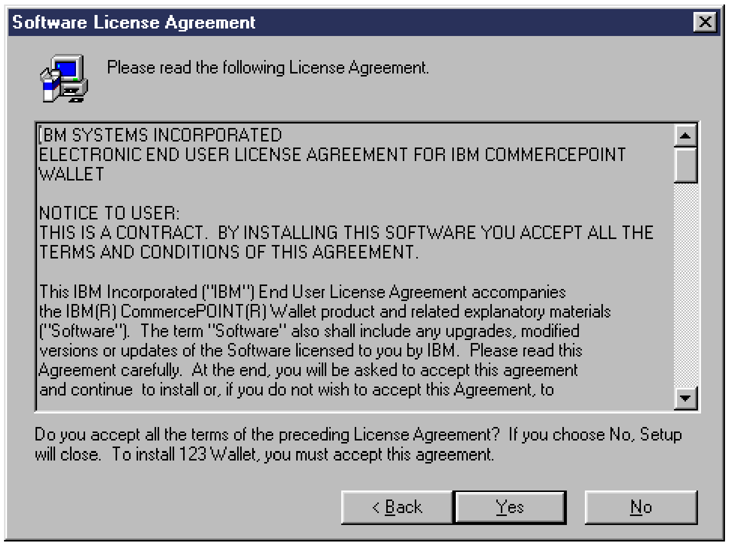
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

Figure 356. IBM Consumer Wallet Setup — Welcome Window

3. From the Software License Agreement window shown in Figure 357 on page 373, click the **Yes** button to accept the license agreement.


Software License Agreement

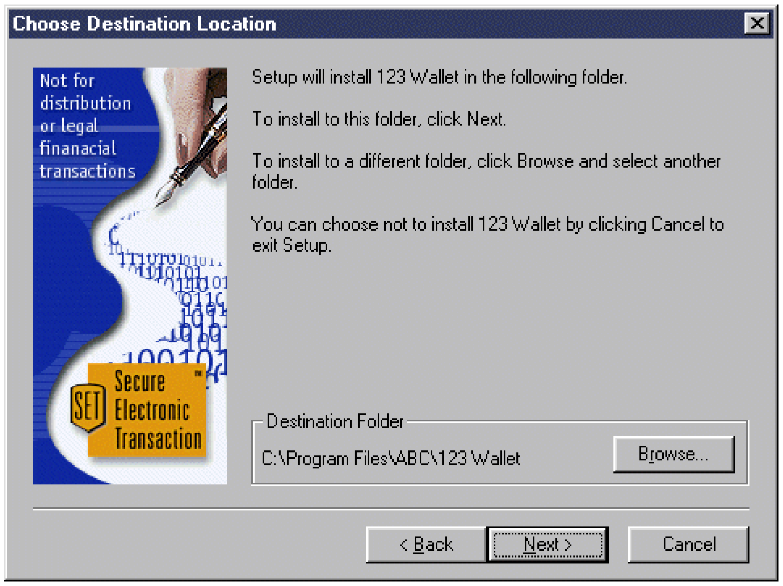
Please read the following License Agreement.

```
IBM SYSTEMS INCORPORATED
ELECTRONIC END USER LICENSE AGREEMENT FOR IBM COMMERCEPOINT
WALLET

NOTICE TO USER:
THIS IS A CONTRACT.  BY INSTALLING THIS SOFTWARE YOU ACCEPT ALL THE
TERMS AND CONDITIONS OF THIS AGREEMENT.

This IBM Incorporated ("IBM") End User License Agreement accompanies
the IBM(R) CommercePOINT(R) Wallet product and related explanatory materials
("Software").  The term "Software" also shall include any upgrades, modified
versions or updates of the Software licensed to you by IBM.  Please read this
Agreement carefully.  At the end, you will be asked to accept this agreement
and continue  to install or, if you do not wish to accept this Agreement, to
```

Do you accept all the terms of the preceding License Agreement? If you choose No, Setup will close. To install 123 Wallet, you must accept this agreement.

Figure 357. IBM Consumer Wallet Setup — Software License Agreement Window

4. From the Choose Destination Location window shown in Figure 358, click the **Next** button to accept default destination location directory.


Choose Destination Location

Setup will install 123 Wallet in the following folder.

To install to this folder, click Next.

To install to a different folder, click Browse and select another folder.

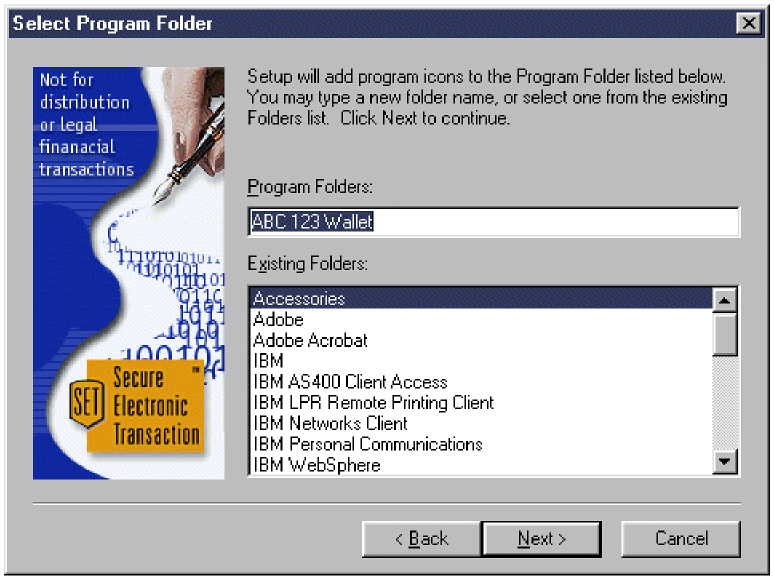
You can choose not to install 123 Wallet by clicking Cancel to exit Setup.

Destination Folder: C:\Program Files\ABC\123 Wallet

Not for distribution or legal finanacial transactions
SET Secure Electronic Transaction
Browse...
< Back
Next >
Cancel

Figure 358. IBM Consumer Wallet Setup — Choose Destination Location Window

5. From the Select Program Folder window shown in Figure 359 on page 374, click the **Next** button to accept the default Program Folder.


Select Program Folder

Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing Folders list. Click Next to continue.

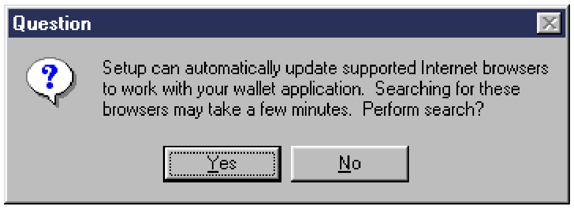
Program Folders:
ABC 123 Wallet

Existing Folders:
Accessories
Adobe
Adobe Acrobat
IBM
IBM AS400 Client Access
IBM LPR Remote Printing Client
IBM Networks Client
IBM Personal Communications
IBM WebSphere

Not for distribution or legal finanacial transactions
Secure Electronic Transaction
SET
< Back
Next >
Cancel

Figure 359. IBM Consumer Wallet Setup — Select Program Folder Window

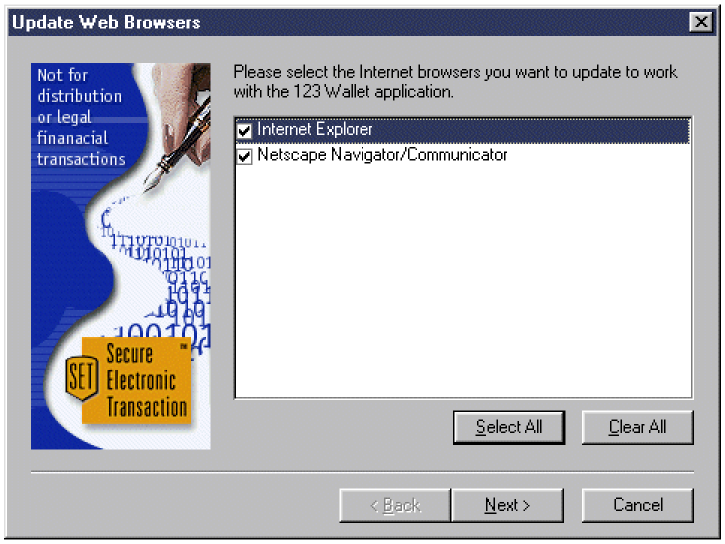
6. From the Question panel shown in Figure 360, click the **Yes** button to allow the setup program to perform a search for supported browsers to update.


Question

Setup can automatically update supported Internet browsers to work with your wallet application. Searching for these browsers may take a few minutes. Perform search?

Figure 360. IBM Consumer Wallet Setup — Perform Search Question Panel

7. From the Update Web Browsers window shown in Figure 361, click the **Next** button to update the default browsers that are selected.


Update Web Browsers

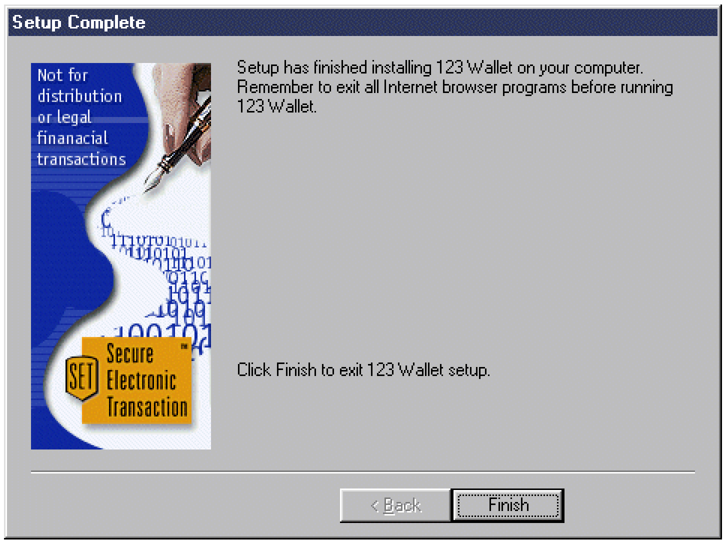
Please select the Internet browsers you want to update to work with the 123 Wallet application.

✓ Internet Explorer
✓ Netscape Navigator/Communicator

Not for distribution or legal finanacial transactions
Secure Electronic Transaction
SET
Select All
Clear All
< Back
Next >
Cancel

Figure 361. IBM Consumer Wallet Setup — Update Web Browser Window

8. From Setup Complete window shown in Figure 362, click the **Finish** button to complete the setup.


Setup Complete

Setup has finished installing 123 Wallet on your computer. Remember to exit all Internet browser programs before running 123 Wallet.

Click Finish to exit 123 Wallet setup.

Not for distribution or legal finanacial transactions
Secure Electronic Transaction
SET
< Back
Finish

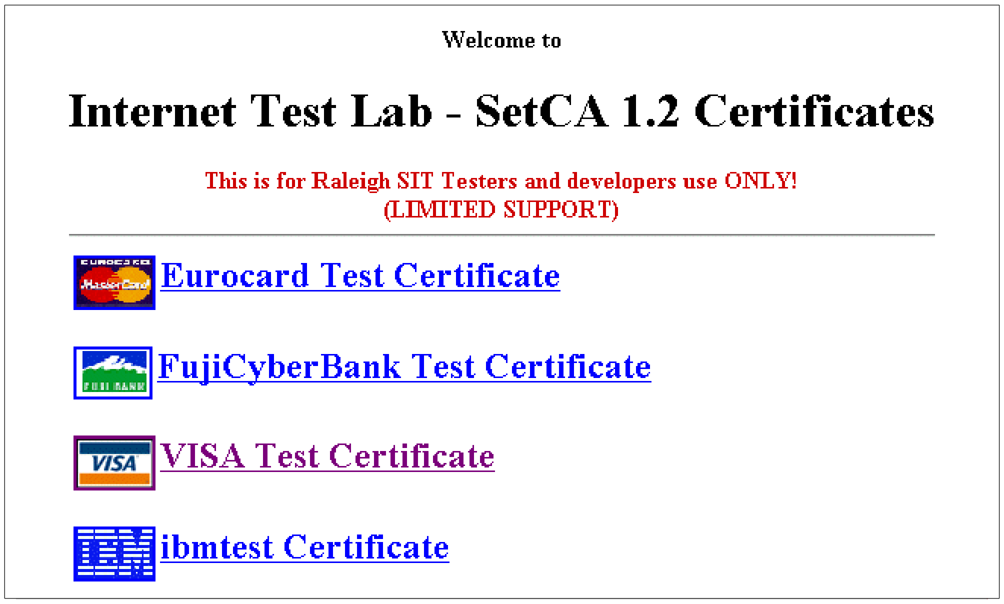
Figure 362. IBM Consumer Wallet Setup — Setup Complete Window

Now, your eWallet is installed and ready to request a SET certificate from your acquirer and load it into your eWallet.

16.4 Getting a SET Certificate for the IBM Consumer Wallet

To show the process of requesting a SET certificate from your acquirer and loading it into your eWallet, we are going to use the IBM SetCA internal test Web site and a test IBM Consumer Wallet. Follow this process:

1. Connect to the acquirer's Web page for certificate requests as shown in Figure 363. Then, click on the URL that points to your credit card.


Welcome to

Internet Test Lab - SetCA 1.2 Certificates

This is for Raleigh SIT Testers and developers use ONLY!
(LIMITED SUPPORT)

Eurocard Test Certificate

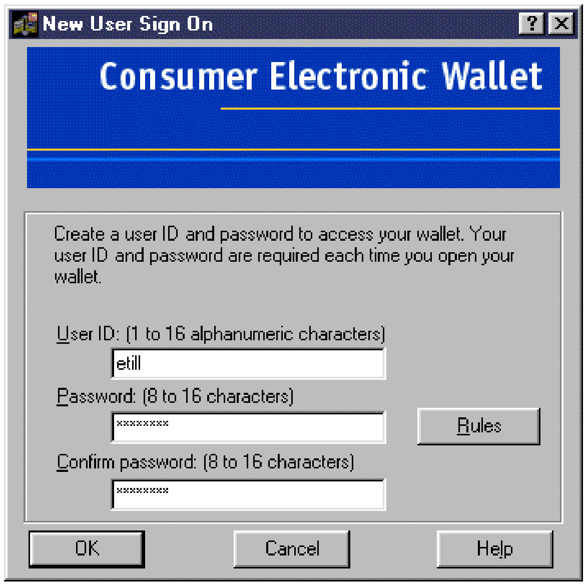
FujiCyberBank Test Certificate

VISA Test Certificate

ibmtest Certificate

Figure 363. IBM Consumer Wallet Certificate Setup — Sample Acquirer Web Page

2. From the New User Sign On window shown in Figure 364, type a user ID, and a password and its confirmation. Click **OK** to create the user and sign on.



Create a user ID and password to access your wallet. Your user ID and password are required each time you open your wallet.

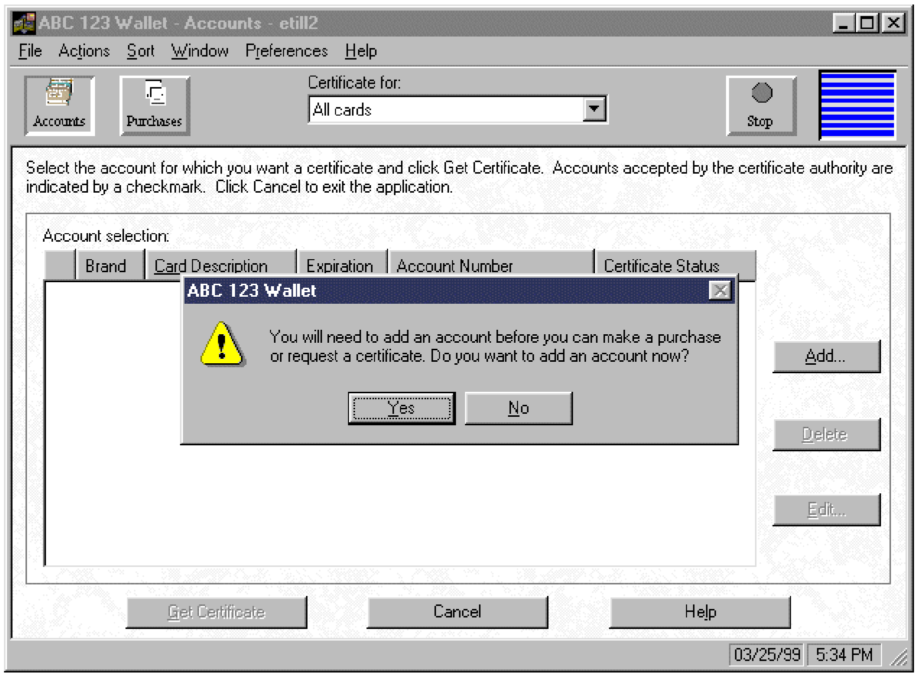
User ID: (1 to 16 alphanumeric characters) etill

Password: (8 to 16 characters) ********

Confirm password: (8 to 16 characters) ********

Figure 364. IBM Consumer Wallet Certificate Setup — New User Sign On Window

3. From the dialog panel of the eWallet window shown in Figure 365, click the **Yes** button to add a new account.


ABC 123 Wallet - Accounts - etill2
File Actions Sort Window Preferences Help
Accounts
Purchases
Stop
Add...
Delete
Edit...
Get Certificate
Cancel
Help

Certificate for: All cards

Select the account for which you want a certificate and click Get Certificate. Accounts accepted by the certificate authority are indicated by a checkmark. Click Cancel to exit the application.

Account selection:

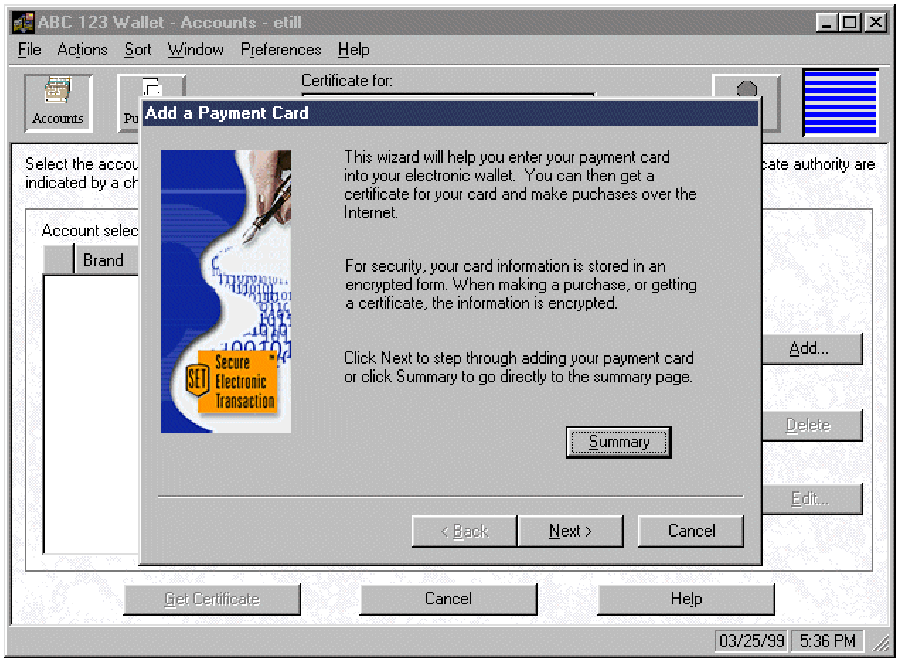
	Brand	Card Description	Expiration	Account Number	Certificate Status

ABC 123 Wallet

You will need to add an account before you can make a purchase or request a certificate. Do you want to add an account now?

Figure 365. IBM Consumer Wallet Certificate Setup — Add Account Window

4. From the Add a Payment Card panel shown in Figure 366 on page 377, click the **Summary** button to complete all required information from a single window.



This wizard will help you enter your payment card into your electronic wallet. You can then get a certificate for your card and make purchases over the Internet.

For security, your card information is stored in an encrypted form. When making a purchase, or getting a certificate, the information is encrypted.

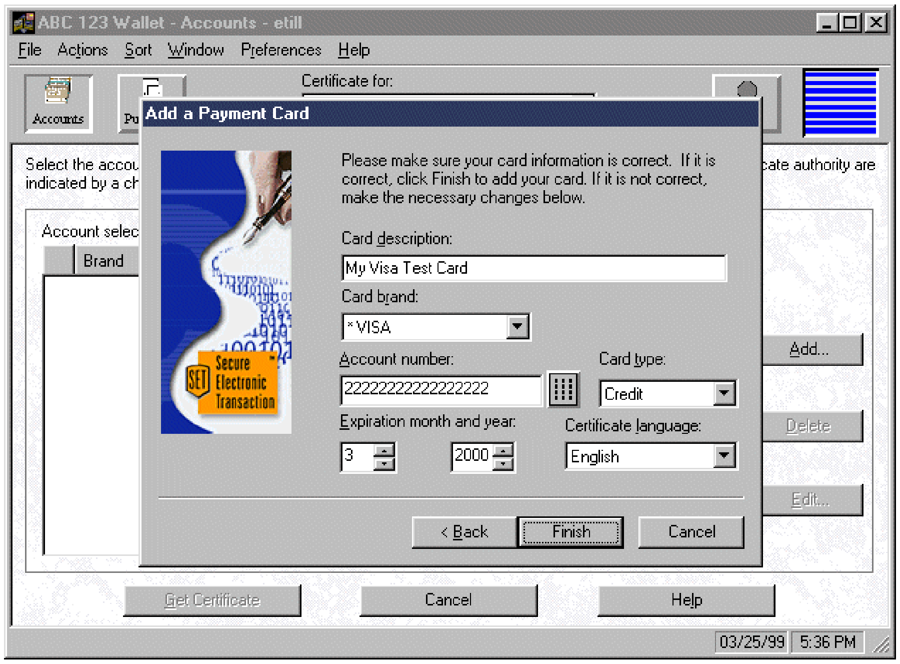
Click Next to step through adding your payment card or click Summary to go directly to the summary page.

Figure 366. IBM Consumer Wallet Certificate Setup — Add Payment Card

5. From the Add a Payment Card summary panel shown in Figure 367 on page 378, complete the following fields with the information provided by your acquirer:

   - Card description
   - Card brand
   - Account number
   - Card type
   - Expiration month and year
   - Certificate language

   Click the **Finish** button to add the payment card.


ABC 123 Wallet - Accounts - etill

Add a Payment Card

Please make sure your card information is correct. If it is correct, click Finish to add your card. If it is not correct, make the necessary changes below.

Card description: My Visa Test Card

Card brand: * VISA

Account number: 2222222222222222

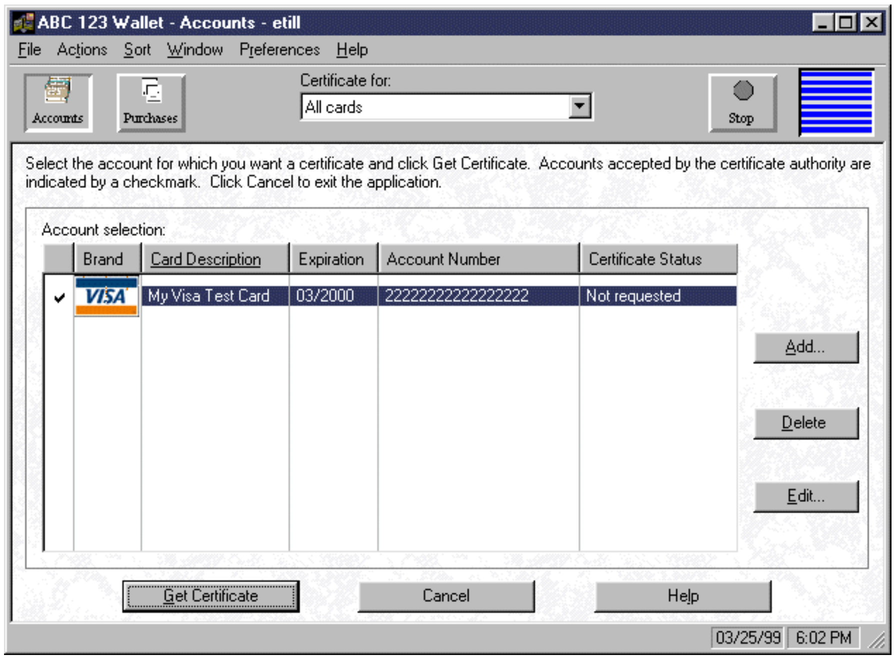
Card type: Credit

Expiration month and year: 3 2000

Certificate language: English

Figure 367. IBM Consumer Wallet Certificate Setup — Add Payment Card Summary

6. From the Accounts view of the eWallet window shown in Figure 368, click the **Get Certificate** button to get the certificate from the acquirer.


ABC 123 Wallet - Accounts - etill

Certificate for: All cards

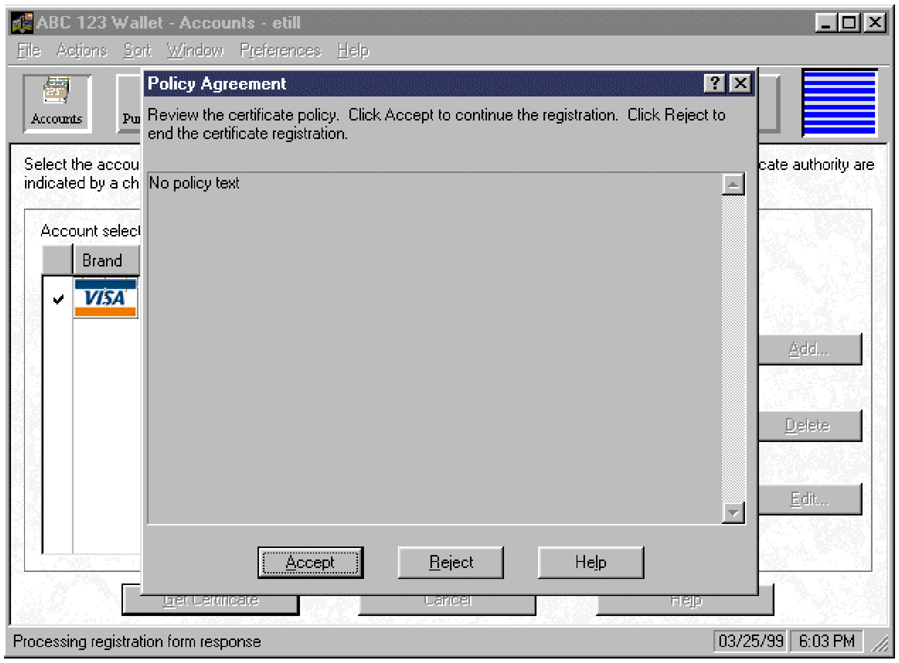
Select the account for which you want a certificate and click Get Certificate. Accounts accepted by the certificate authority are indicated by a checkmark. Click Cancel to exit the application.

Account selection:

	Brand	Card Description	Expiration	Account Number	Certificate Status
✓	VISA	My Visa Test Card	03/2000	2222222222222222	Not requested

Figure 368. IBM Consumer Wallet Certificate Setup — Accounts View Window

7. From the Policy Management panel shown in Figure 369 on page 379, read the policy agreements of your acquirer. Click the **Accept** button to accept them.


ABC 123 Wallet - Accounts - etill
File Actions Sort Window Preferences Help
Accounts
Policy Agreement
Accept
Reject
Help
Add...
Delete
Edit...
Brand

Review the certificate policy. Click Accept to continue the registration. Click Reject to end the certificate registration.

No policy text

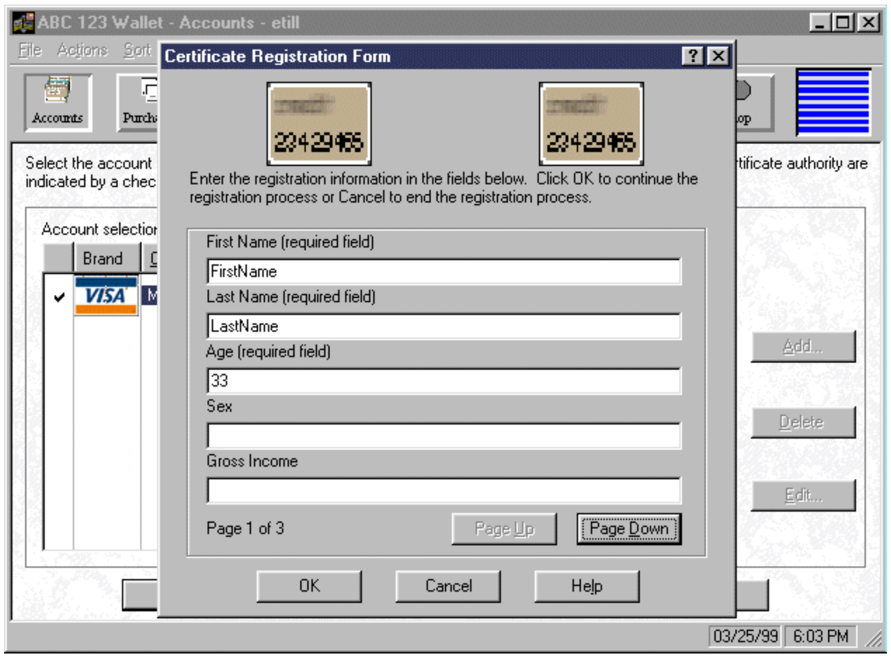
Processing registration form response 03/25/99 6:03 PM

Figure 369. IBM Consumer Wallet Certificate Setup — Policy Agreement Panel

8. From the Certificate Registration Form page 1 shown in Figure 370, complete the following required fields:

   - First name
   - Last name
   - Age

   Click **Page Down** to go to page 2.



Enter the registration information in the fields below. Click OK to continue the registration process or Cancel to end the registration process.

First Name (required field): FirstName
Last Name (required field): LastName
Age (required field): 33
Sex:
Gross Income:

Page 1 of 3

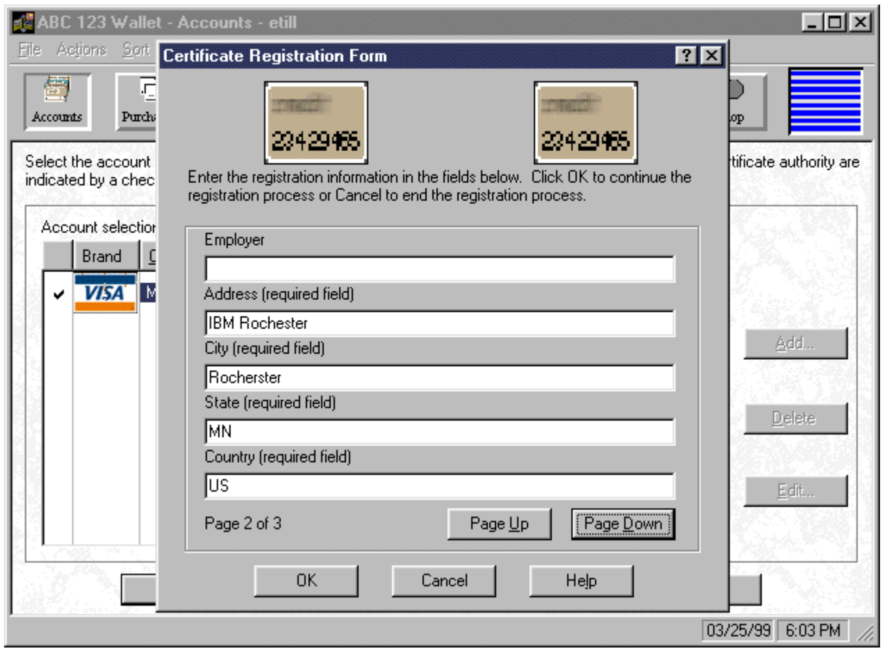
03/25/99 6:03 PM

Figure 370. IBM Consumer Wallet Certificate Setup — Certificate Registration (Page 1)

9. From the Certificate Registration Form page 2 shown in Figure 371, complete the following required fields:

   - Address
   - City
   - State and country

   Click **Page Down** to go to page 3.



ABC 123 Wallet - Accounts - etill

Certificate Registration Form

Enter the registration information in the fields below. Click OK to continue the registration process or Cancel to end the registration process.

Employer:

Address (required field): IBM Rochester

City (required field): Rocherster

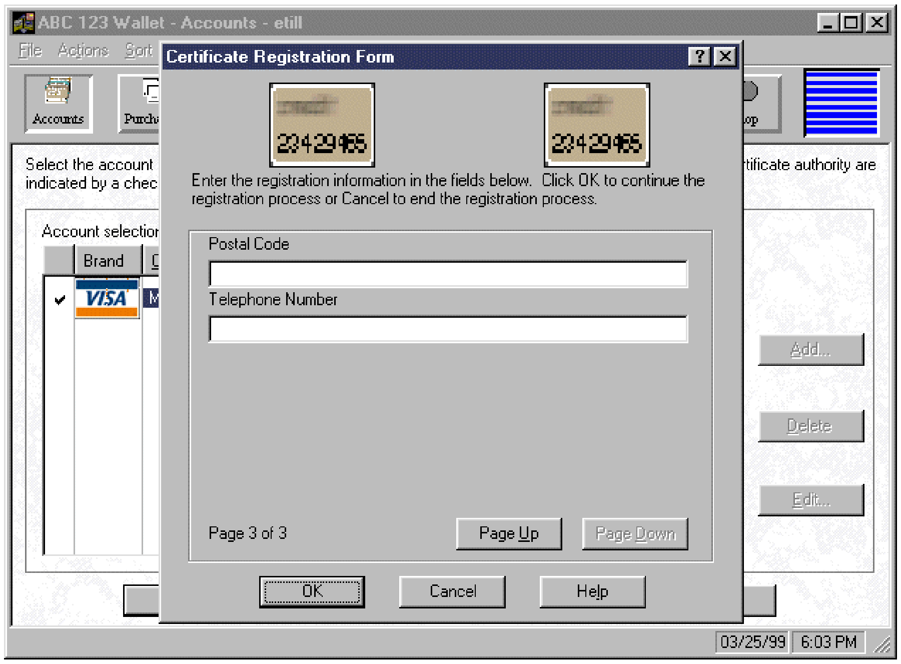
State (required field): MN

Country (required field): US

Page 2 of 3

Figure 371. IBM Consumer Wallet Certificate Setup — Certificate Registration (Page 2)

10. From the Certificate Registration Form page 3 shown in Figure 372 on page 381, click **OK** to send the registration form.



Enter the registration information in the fields below. Click OK to continue the registration process or Cancel to end the registration process.

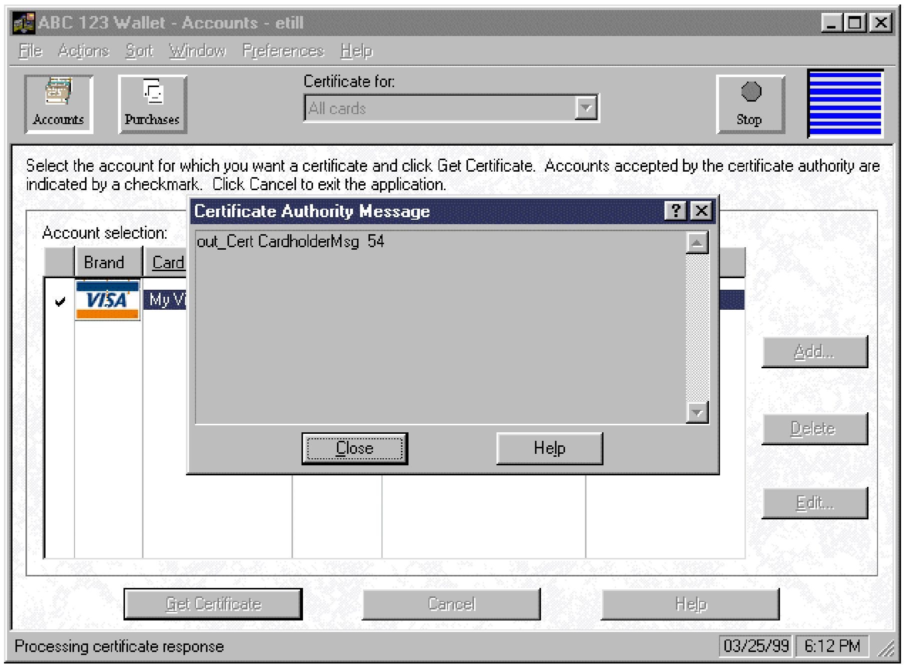
Postal Code

Telephone Number

Page 3 of 3

Figure 372. IBM Consumer Wallet Certificate Setup — Certificate Registration (Page 3)

11.From the Certificate Authority Message panel shown in Figure 373, click the **Close** button.



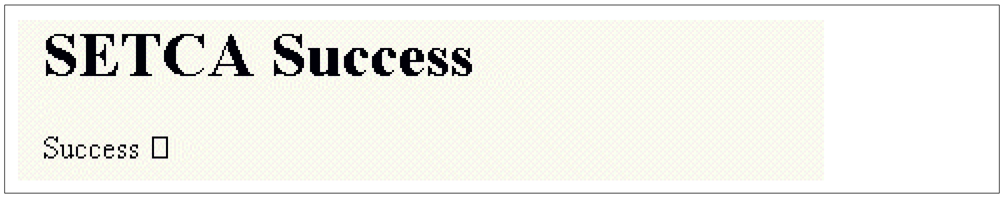
Select the account for which you want a certificate and click Get Certificate. Accounts accepted by the certificate authority are indicated by a checkmark. Click Cancel to exit the application.

out_Cert CardholderMsg 54

Processing certificate response

Figure 373. IBM Consumer Wallet Certificate Setup — Certificate Authority Message

The eWallet program closes and a success certificate setup message is displayed in the acquirer Web page, as shown in Figure 374 on page 382.


SETCA Success

Success □

Figure 374. IBM Consumer Wallet Certificate Setup — Success Certificate Setup Page

Now your eWallet is ready to use the SET protocol to authorize credit card transactions with your acquirer.

Chapter 17. Interfacing to Our Back-End Business System

This chapter explores the integration of our ShopITSO store with an existing back-end system. In our example, we simplified some of the data mapping by ensuring that our back-end system had all of the necessary data. Differences in the data map between the Net.Commerce database and the back-end system can be resolved in the APIs that interface the two systems.

17.1 Description of Our Example

There are four integration issues that we had to deal with in our site design example:

- Integration with the order process so that Net.Commerce retrieves the product price using the back-end pricing mechanism. Orders originated from the Net.Commerce site are passed to the back-end system for fulfillment. The back-end system must also e-mail the order confirmation to the client. The back-end system also e-mails the order confirmation to the client.
- Integration with the Net.Commerce database and cache mechanism to propagate changes in the back-end product information to Net.Commerce product related tables and to purge outdated product pages from the Net.Commerce cache.
- Integration with the back-end shipping module to request capture from the acquirer upon order fulfillment.
- Initial data load from the back-end system tables to the Net.Commerce database. This process is described in Chapter 15, “Importing Business Data into Net.Commerce” on page 319.

This chapter describes the way that we implemented our solution to these specific back-end integration issues. It is important to understand that there may be other integration issues that you may have to manage. For example, consider these cases:

- Loading current customer data to Net.Commerce shopper tables in case your site works with registered users.
- Implementing an overridable function to check the inventory on the back-end system before order approval.
- Giving the shopper the possibility to query the order status directly from the back-end system.
- Enabling data propagation from the Net.Commerce tables to the back-end system tables.

17.2 The Pricing and Orders Process

Figure 375 on page 384 describes the pricing and order integration between Net.Commerce and our back-end system.

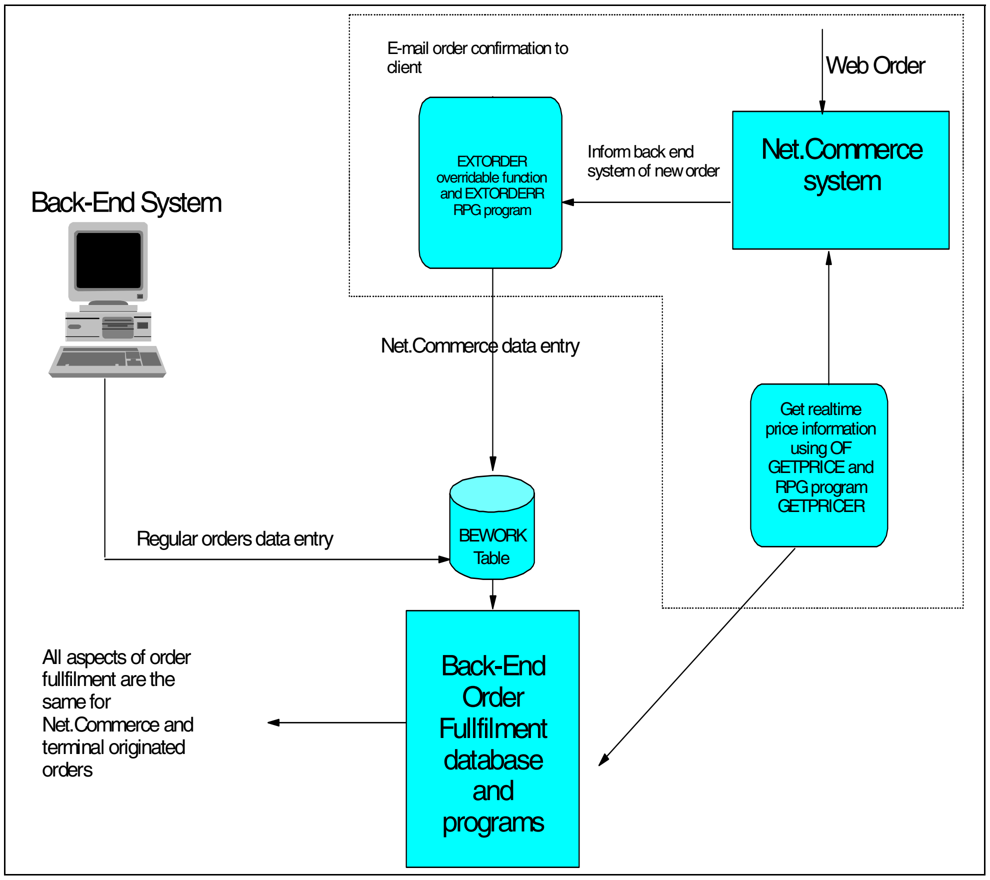

E-mail order confirmation to client
Web Order
Net.Commerce system
Inform back end system of new order
EXTORDER overridable function and EXTORDERR RPG program
Back-End System
Net.Commerce data entry
Get realtime price information using OF GETPRICE and RPG program GETPRICER
Regular orders data entry
BEWORK Table
All aspects of order fullfilment are the same for Net.Commerce and terminal originated orders
Back-End Order Fullfilment database and programs

Figure 375. Back End System Integration — Order Flow and Pricing

The existing back-end system receives orders from a workstation order entry application. After the user completes the order, the order request is written by the existing application to a BEWORK order table. The existing application also has an existing API to retrieve the product price. This API is implemented by the RPG program GETPRICER. The price retrieval process is described in detail in Chapter 19, “Implementing Overridable Functions” on page 413. The back-end system has a fully functional order fulfillment process that reads the BEWORK table and processes the orders from it.

When a user enters the Net.Commerce shop, they must see the correct prices for the products for which they are looking. In our site example, the only source for the product price is the back-end pricing system. In the site implementation, we changed the tasks GET_BASE_UNIT_PRC and GET_BASE_SPE_PRC to use our overridable function GETPRICE. GETPRICE is called by Net.Commerce. It gets some parameters and invokes the back-end system GETPRICER API. The GETPRICE overridable function is described in Chapter 17, “Interfacing to Our Back-End Business System” on page 383.

After the user places the purchase order, use the back-end system order fulfillment mechanism to fulfill the order. The Net.Commerce command OrderProcess is used to place orders from shoppers, among other tasks, the OrderProcess call task EXT_ORD_PROC. In our site implementation, we changed the task EXT_ORD_PROC to use our overridable function EXTORDER. Our function places an entry in a data queue for later processing and returns control to Net.Commerce. The data queue is processed by RPG PROGRAM EXTORDERR, which sends e-mail confirmation to the customer and inserts a row to the order work table BEWORK. After the new order is placed in the BEWORK table, the back-end system processes the order in exactly the same way it processes the orders from the back-end system order entry application.

Figure 376 describes the programs involved in the order process.

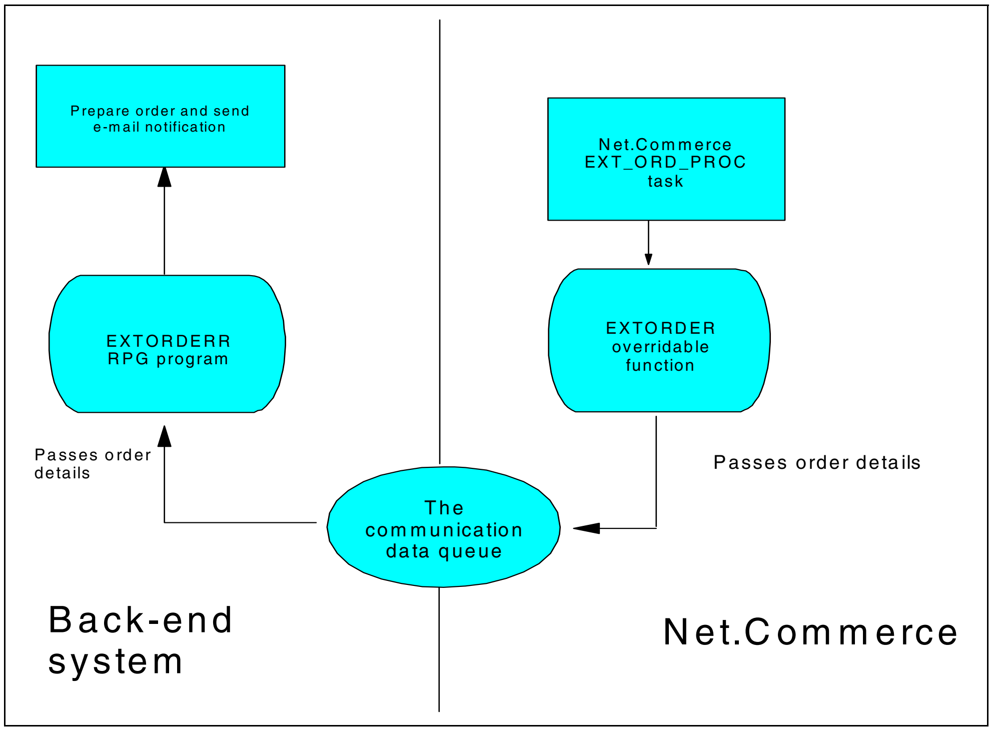

Prepare order and send e-mail notification
Net.Commerce EXT_ORD_PROC task
EXTORDERR RPG program
EXTORDER overridable function
Passes order details
Passes order details
The communication data queue
Back-end system
Net.Commerce

Figure 376. Process Order from the Net.Commerce Site

Figure 377 on page 386 and Figure 378 on page 387 display the EXTORDER overridable function source code. See Chapter 19, “Implementing Overridable Functions” on page 413, for more details on overridable functions.

```
//*********************************************************/
// This function will inform the back end system of the    */
// arrival of new order. The function will place some basic */
// details in a data queue object.                       */
//                                                     */
// It will be called by the EXT_ORD_PROC task            */
//                                                     */
// Author: Shahar mor                                       */
// Provided AS IS                                           */
//*********************************************************/


#ifdef AS400
       #include "coibm.h"
#endif

#include <qsnddtaq.h>          // Data queue          (1)
#include "objects/objects.pch" // Net.Commerce include

// The following define is for compatibility with other platform compilers

#if defined(WIN32)
  #define __DLL_EXPORT__ __declspec(dllexport)
#else
  #define __DLL_EXPORT__
#endif


// Handling the trace option. Un remark the next line to enable trace.
//#define __TRACE_ExtOrder__
#ifdef __TRACE_ExtOrder__
 typedef TraceYes Trace;
#else
 typedef TraceNo Trace;
#endif

#define Q_LEN  360   // Data queue entry length


static Trace trace ("GetPrice ("__FILE__")");

typedef struct Q_entry {                        (2)
     char RefNumber[15];
     char MerchNumber[30];
     char ShopperId[31];
     char TotalPrice[15];
     char TotalTax[15];
     char Email[254];
} Q_entry; // Data queue entry structure

class __DLL_EXPORT__ ExtOrder : public NC_OverridableFunction
{
  public:
    ExtOrder() { }

    virtual  ~ExtOrder() { }

     void operator delete (void *p){ ::delete p; }
```

Figure 377. The EXTORDER Source (Part 1 of 2)

```
// Handle failed registration */

  virtual void FailedRegistration (NC_RegistrationID &RegID, const ErrorMsg_Reg *Err)
    {
     error << indent << "Error : ExtOrder Registration failed" << endl;
    }
// Main function */

  virtual bool Process (const HttpRequest &Req, HttpResponse &Res, NC_Environment &Env)
    {

      Q_entry BackEndSystem;    // Queue structure
      String Stmt;
      Row SqlRow;
      char Test[300];

// Define the environment
     static const StringWithOwnership _PARAM_NAME_ORDER_REF_NUM("ORDER_REF_NUM");

// Get function parameters from the env.
// 1. The order reference number

      String* OrderRefNum = (String*) Env.Seek(_PARAM_NAME_ORDER_REF_NUM);
       if (OrderRefNum == NULL)
        {
            error << indent << "Error : Cant get order   number    " << endl;
             return false;
        }
         String stmt;                        (3)
       stmt << "SELECT trim(char(ORRFNBR)), ORMORDER, ORPRTOT,";
        stmt << "ORTXTOT, SHLOGID , trim(SAEMAIL1), SARFNBR FROM ORDERS, SHOPPER ,SHADDR
";
       stmt << "WHERE ORRFNBR =" << *OrderRefNum;
       stmt << " AND ORSHNBR = SASHNBR";
       stmt << " AND ORSHNBR = SHRFNBR";
       stmt << " ORDER BY SARFNBR DESC";
       strcpy(Test,stmt.c_str());

      SQL Sql(*(DataBaseManager::GetCurrentDataBase()), stmt);

       if (Sql.getNextRow(SqlRow) != ERR_DB_NO_ERROR)
        {
             return false;
        }

// Send the data queue entry

        memset(&BackEndSystem,' ',sizeof(BackEndSystem));          (4)
      strcpy(BackEndSystem.RefNumber,SqlRow.getCol(1).c_str());
      strcpy(BackEndSystem.MerchNumber,SqlRow.getCol(2).c_str());
      strcpy(BackEndSystem.TotalPrice,SqlRow.getCol(3).c_str());
      strcpy(BackEndSystem.TotalTax,SqlRow.getCol(4).c_str());
      strcpy(BackEndSystem.ShopperId,SqlRow.getCol(5).c_str());
      strcpy(BackEndSystem.Email,SqlRow.getCol(6).c_str());

       QSNDDTAQ("TESTQ     ",       /* dtaq name       */
        "NETCBE    ",        /* dtaq lib        */                 (5)
       Q_LEN ,            /* length of data  */
       &BackEndSystem);    /* data       */

      return true;
}
};
static bool X2 = NC_ApiManager::GetUniqueInstance().RegisterApi("IBM", "NC",
                                "ExtOrder", 1.0, new ExtOrder);
```

Figure 378. The EXTORDER Source (Part 2 of 2)

Consider these remarks about the EXTORDER program:

- Since we use AS/400 data queue support, we must include the data queue header file in our source.

- The Q_entry structure describes the data that we will send to the communication data queue. The exact same structure will be used by the EXTORDERR program.
- The OF uses Net.Commerce database classes to select data from the database. In our example, we prepare a select statement that retrieves order details from different Net.Commerce tables. You can retrieve the database values in the RPG program EXTORDERR. However, you may find it easier to use the Net.Commerce database classes.
- The returned data from the tables is fetched by using the Net.Commerce database classes.
- The returned data is sent to the data queue. The RPG program EXTORDERR receives the data from the data queue and continues the process. Net.Commerce receives positive response from our function. The shopper receives positive response on their browser unless for some reason our select statement failed to fetch the order details from the Net.Commerce database.

Figure 379 through Figure 382 on the following pages list the RPG EXTORDERR program that is used to process the data queue entries sent by the EXTORDER overridable function.

```
H*********************************************************************
H* This program will wait on data queue for notification on new      *
H* orders from the Net.Commerce web site. It will then be able to    *
H* activate the order fulfillment process in the back end system and *
H* optionally send e-mail confirmation to the customer.             *
H*                                                                  *
H* Entries in the data queue will be placed by the overridable       *
H* function extorder.                                                *
H* Author: Shahar mor                                                *
H* Provided AS IS                                                    *
H*********************************************************************
H*
H DFTACTGRP(*NO) ACTGRP(*CALLER) BNDDIR('QC2LE')   (1)
F*
FBEWORK   O    E           DISK    RENAME(BEWORK:RBEWORK)
F*
DDqEntry         DS                        (2)
D  RefNumber                  16
D  MerNumber                  31
D  ShopperId                  32
D  TotalPrice                 16
D  TaxPrice                   16
D  ShoppEmail                255
D*
DReturnLen       S             5P 0
DTrimChar        S            15A
DMsg             S           512A
D*
DForEver         C                 CONST(-1)
DDataQueue       C                 CONST('TESTQ')
DDataqLib        C                 CONST('NETCBE')
DEndProg         C                 CONST('000000000000000')
DNetCommerce     C                 CONST('Net.Commerc')
DProcess         C                 CONST('1')
DEndLine         C                 CONST(X'0D')
D*
D* Prototype external QRCVDTAQ program
D Rcvdtaq        PR                EXTPGM('QRCVDTAQ')
D   QueueName                 10A   CONST
D   QueueLib                  10A   CONST
D   DtaLen                     5P 0
D   Dta                     3000A   OPTIONS(*VARSIZE)
D   WaitTime                   5P 0 CONST
D*
D* Prototype internal check e mail validity routine
DIsEmail         PR            1N
D EmailAddr                  254A   CONST
D*
D* Prototype internal null removal
DRmvNull         PR          200A
D Dta                        200A   CONST OPTIONS(*VARSIZE)
D StrLen                       5P 0 VALUE
D*
D* Prototype external c function
Datof            PR            8F   EXTPROC('atof')   (3)
D StringPtr                     *   VALUE
D*
C*
C***********************************************************
C* Main program logic
C***********************************************************
C*
C                   EXSR      GetEntry
C                   DOW       RefNumber <> EndProg
C                   EXSR      BackEnd
C                   IF        IsEmail(ShoppEmail)
C                   EXSR      SendMail
C                   ENDIF
C                   EXSR      GetEntry
C                   ENDDO
C*
C                   EVAL      *INLR = *on
```

Figure 379. The EXTORDERR Program Source (Part 1 of 3)

```
C******************************************************************
   C    GetEntry     BEGSR
  C******************************************************************
   C*
   C* Recieve the next entry from the Net.Commerce site
   C*
   C                 CALLP    RcvDtaq(DataQueue:
   C                                 DataqLib:
   C                                 ReturnLen:
   C                                 DqEntry:
   C                                 ForEver)
   C*
   C                 ENDSR
  C******************************************************************
   C    SendMail     BEGSR         (4)
  C******************************************************************
   C*
   C*   Prepare and send e-mail confirmation.
   C*
   C                 EVAL     Msg = 'Dear customer,' + EndLine
   C                           + 'Thank you for Buying in our shop'
   C                           + EndLine + 'Your order number is:'
   C                           + %trim(RmvNull(MerNumber:31))
   C                           + EndLine + 'Your customer number is:'
   C                           + %trim(RmvNull(ShopperId:32))
   C                           + EndLine +  'please visit our site '
   C                           + 'at www.redbooks.ibm.com to check '
   C                            + 'your order status'
   C*
   C                 CALL     'ORDERC'
   C                 PARM                  Msg
   C                 PARM                  ShoppEmail
   C*
   C                 ENDSR
  C******************************************************************
   C    BackEnd      BEGSR                  (5)
  C******************************************************************
   C*
   C* Integrate with back end system is done here(Write to work file)
   C*
   C                 MOVE     *DATE        BODATE
   C                 TIME                  BOTIME
   C                 EVAL     BOAFLG = Process
   C                 EVAL     BOWS = NetCommerce
   C                 EVAL     BOONUM = %trim(RmvNull(MerNumber:31))
   C                 EVAL     TrimChar = RmvNull(TotalPrice:16)
   C                 EVAL     BOSUM = atof(%ADDR(TrimChar))
   C                 WRITE    RBEWORK
   C
   C                 ENDSR
  C*********************************************************************
   PIsEmail         B
   DIsEmail         PI            1N
   D EmailAddr                  254A   CONST
   C*
   DValid           S             1N
   DApos            S             3P 0
   DDpos            S             3P 0
   DLpos            S             3P 0
   DI               S             3P 0
   DJ               S             3P 0
   C*
   C*  Will Check for email validity
   C*
   C                 IF       EmailAddr = *BLANK
   C                 RETURN    *OFF
   C                 ENDIF
   C*
   C                 EVAL     Valid = *ON
   C    ' '          CHECKR   EmailAddr    Lpos
   C                 EVAL     Apos = %scan('@':EmailAddr)
   C*
```

Figure 380. The EXTORDERR Program Source (Part 2 of 3)

```
C* Find last occurance of the decimal point character
C                   EVAL      I = 1
C                   EVAL      J = 1
C                   DOW       J > 0
C                   EVAL      J = %scan('.':EmailAddr:I)
C                   IF        J <> 0
C                   EVAL      I = J+1
C                   ENDIF
C                   ENDDO
C                   EVAL      Dpos = I - 1
C*
C                   IF        Apos < 1 or
C                             (Dpos - Apos) < 2  or
C                             (Lpos - Dpos) > 3  or
C                             (Lpos - Dpos) < 2
C                   EVAL      Valid = *OFF
C                   ENDIF
C*
C                   Return    Valid
C*
PIsEmail          E
C*********************************************************************
PRmvNull          B      (6)
D*
DRmvNull          PI           200A
D Dta                          200A   CONST OPTIONS(*VARSIZE)
D StrLen                         5P 0 VALUE
DReturnStr        S            200A
DNull             C                   CONST(X'00')
C*
DI                S              3P 0
C*
C                   EVAL      I = 1
C                   EVAL      ReturnStr = %subst(dta:1:StrLen)
C                   DOW       I  <= Strlen
C                   IF        %subst(ReturnStr:I:1) = Null
C                   EVAL      %subst(ReturnStr:I:1) = *BLANK
C                   ENDIF
C                   EVAL      I = I + 1
C                   ENDDO
C*
C                   Return    ReturnStr
C*
PRmvNull          E
```

Figure 381. The EXTORDERR Program Source (Part 3 of 3)

Consider these remarks in regard to the EXTORDERR program:

- The program must use the QC2LE binding directory since it will use the C function to convert the character string to a number. Using C routines is the recommended method of converting string to numeric representation. However, nothing prevents you from using your own internal routines for this task. The conversion is needed since the C++ overridable function sends numeric values as strings to the data queue.
- The Dqentry structure describes the data queue entry. The data queue entry is placed in the data queue by the overridable function EXTORDER, which Net.Commerce calls from the EXT_ORD_PROC task.
- Map the external C function to convert strings to doubles. All entries in the data queue are placed as strings. Numeric fields, such as total price, should be converted to numeric representation before processing.
- The e-mail integration sends simple confirmation. Program ORDERC, listed in Appendix A, “Source Code Samples” on page 463, actually sends the e-mail message.

- The program inserts a row to the BEWORK table. The back-end application uses this table to process orders. The row contains information that was sent to the data queue by the overridable function EXTORDER.
- The string from C++ contains NULLS(X'00'). To perform string operations in RPG, we use the internal RmvNull procedure to remove all nulls from the data queue strings before processing.

In summary, Net.Commerce uses our own functions to calculate the price and perform order fulfillment. Our functions are used as a gate between the Net.Commerce frame to the back-end business process.

17.3 Table Synchronization and Cache Mechanism

Since we use the back-end system to maintain the product and pricing information, we must consider these points:

- Every change in product information on the back-end system should be reflected in the Net.Commerce system. For example, if a product was added to the back-end system, it should be automatically added to the Net.Commerce database. If a product description was changed in the back-end system, it must also change in the Net.Commerce database.
- Net.Commerce uses a caching mechanism for storing product details. Whenever the back-end system price is changed, we must signal Net.Commerce that the cached page is no longer reflecting the correct price and should be purged from the cache.

Figure 382 displays the synchronization mechanism that we used to handle these issues.

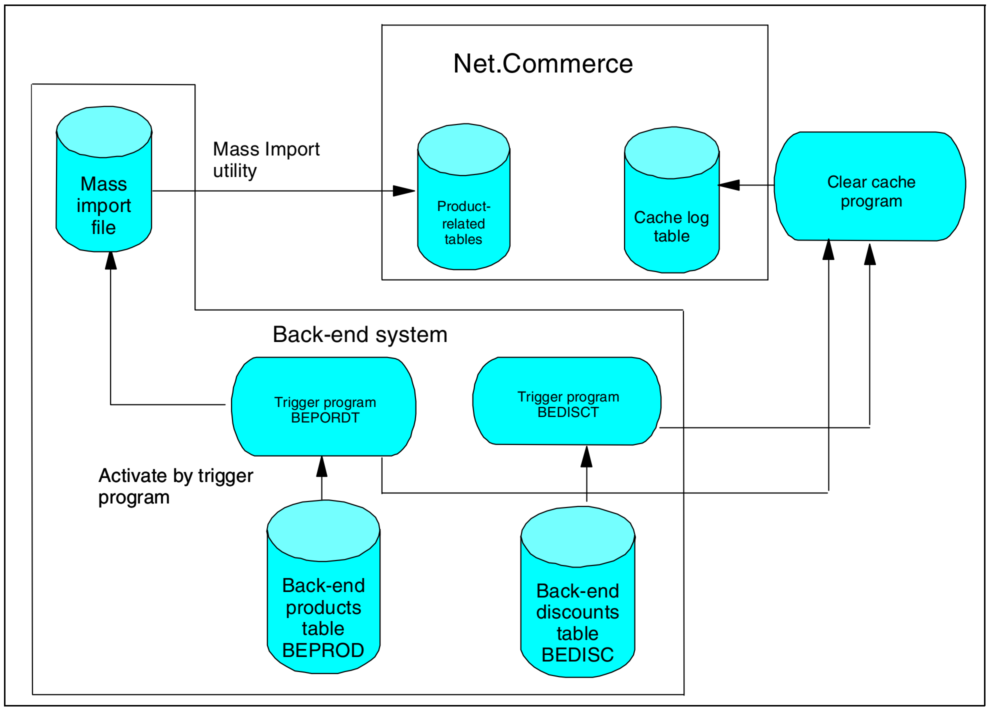


Figure 382. Back-End System Table Synchronization and Cache Handling

17.3.1 Product Information Synchronization

We developed the LOADPRD command described in Chapter 15, “Importing Business Data into Net.Commerce” on page 319, to perform initial data mapping and loading from the back-end system. The LOADPRD utility also supports the importing of one specific product detail.

To synchronize the product information from the back-end system with the product related tables in Net.Commerce, we wrote a trigger program BEPRODT and connected it to the back-end system product table BEPROD. The synchronization process follows this sequence:

1. The trigger program is called after the row insertion or is updated to the back-end products table BEPROD.
2. The trigger program calls the LOADPRD command for the specific changed product. The LOADPRD command adds rows to a mass import file.
3. The mass import file is used by Net.Commerce mass import utility to update Net.Commerce product-related tables. The mass import processes the import file to reflect the changes in the back-end system product information. We can run the mass import once a day or at any frequency we choose depending on our business requirements.

The trigger program BEPRODT is described in Chapter 15, “Importing Business Data into Net.Commerce” on page 319.

17.3.2 Integration with Net.Commerce Cache Mechanism

Net.Commerce uses the caching mechanism for displaying products details. In our example, Net.Commerce retrieves the prices from the back-end system. Therefore, it has no way of knowing that a cached product page no longer reflects the current price and it may serve outdated information from the cache.

To explain our solution for this problem, we must shortly describe the Net.Commerce cache system principals and capabilities.

17.3.2.1 Net.Commerce Cache Mechanism

When a shopper clicks on a link to view a product or category page, only small amounts of system time and resources are actually spent within the server. Most of the time is spent parsing the HTTP request, accessing the database, and dynamically creating the HTML page that the shopper wants to see. Heavy site traffic and a large number of product and category entries in the database can further increase the time it takes for pages to load.

Most HTTP requests on the server are for product and category pages, which are dynamically created by the ProductDisplay and CategoryDisplay commands, respectively. These commands retrieve information from your database, and display the information as an HTML page that was generated from a Net.Data macro. If your product and category information did not change since a page was last viewed, then it should not be necessary for the page to be dynamically re-created the next time a shopper requests it. Serving an equivalent "static" page that has been stored in a cache would be faster. The scope of the caching methods provided by Net.Commerce is mall-wide. If you have a mall (or instance) that contains numerous stores, and you enable and configure caching, the caching will be turned on and work the same for all the stores in that mall (instance). Also, only the site administrator can configure and activate caching.

Store administrators who only have store-level authority cannot configure or activate caching.

Files are stored in the cache on demand. This means that they are not stored in the file system until requested. Only those pages that are created by the commands ProductDisplay and CategoryDisplay are cached by the Net.Commerce caching methods. These commands typically query the PRRFNBR (product reference number) column in the PRODUCT table and the CGRFNBR (category reference number) column in the CATEGORY table. They also query their matching merchant reference numbers from the PRMENBR and CGMENBR columns.

If the file corresponding to the page being accessed is not in the cache file storage, it is generated dynamically in the usual way. The CategoryDisplay or ProductDisplay command calls Net.Data, Net.Data accesses the database and creates the HTML file. Then, the command returns the results, additional parameters are collected, and the final page is sent to the shopper's browser. This HTML page is then stored in the cache, and does not have to be regenerated until the data on which it is based is modified.

If you change information in the database (such as a product price), make sure that any page in the cache that contains the changed information is deleted. Also verify that the page is dynamically re-generated when the product is next viewed by a shopper picking up the new information. This is done for you by the synchronization daemon. When you change certain product or category information in the database, a log record is added to the CACHLOG table. The synchronization daemon queries this table at regular, specified intervals, identifies which HTML pages are affected by the changes, and purges (deletes) them from the cache. This ensures that shoppers will not access any pages that are outdated or incomplete.

For pages to be purged, the synchronization daemon must be properly configured and running. It is not necessary to create any new triggers to purge outdated product or category pages that were generated from the supplied product and category tables. However, you may need to create a custom trigger if you used any external methods that affect the product information page. For example, suppose you created a table, named PRODEXTINFO, that contains extra text information about products to be included in the displayed pages. Suppose the table contains a column, named PEPRNBR, that is a foreign key to the product reference number. It also contains another column, named PETEXT, that contains the text itself. Because column PETEXT is selected in an SQL query by the product display macro, a cache file created from information retrieved must be purged when the PETEXT value for a product changes. When you update the record in PETEXT for PEPRNBR=10, the custom trigger that you create on this table logs the following record to the CACHLOG table:

```
"insert into CACHLOG values ('prrfnbr',10,timestamp)"
```

Now, when the synchronization daemon accesses the CACHLOG table, it discovers a new row and deletes all product pages pertaining to the product with reference number 10. To learn more about the Net.Commerce caching, please refer to the Net.Commerce online help.

17.3.2.2 Integration with Net.Commerce Cache

The product pages displayed by Net.Commerce use price retrieval from the back-end system. In our system, the product price can be changed either by a price change in the BEPROD table or by a change in the discounts table BEDISC.

We developed a command CLRCACH that allows us to signal Net.Commerce of price changes. The command source is available on Appendix A, “Source Code Samples” on page 463. The command is called in the following events:

- The trigger program, on updates in the products table BEPROD, detects change in the product price.
- The trigger program, on insert, update, and delete in the discounts table BEDISC, detects changes in the discounts table.

The command simply inserts a row with the product number in the cachelog table of Net.Commerce. The synchronization daemon of Net.Commerce detects the new row and purges the product details from the cache. The next time a shopper asks for the product information, Net.Commerce retrieves the product price from our back-end system using the GETPRICE overridable function.

Figure 383 on page 396 and Figure 384 on page 397 display the source code for the discount table trigger file.

```
H*********************************************************************
   H*  This trigger program will be called after insert/update operation *
    H*  to the back end system discount table.                         *
    H*                                                                *
   H*  This program will be called on update event only if the update   *
    H*  actually changed anything (ADDPFTRG TRGUPDCND(*CHANGE))            *
    H*  Author: Shahar mor                                                 *
    H*  Provided AS IS                                                 *
  H*********************************************************************
    D*
    D*  The General buffer for trigger programs
    D*
    D Parm1          DS
    D Filenm                1     10
    D Libnm                11     20
    D Mbrnam               21     30
    D Trgevn               31     31
    D Trgtim               32     32
    D Trglvl               33     33
    D Trgreserv            34     36
    D Trgccsid             37     40B 0
    D Trgreserv2           41     48
    D Oldofs               49     52B 0
    D Oldlen               53     56B 0
    D Omapof               57     60B 0
    D Omapln               61     64B 0
    D Newofs               65     68B 0
    D Newlen               69     72B 0
    D Nmapof               73     76B 0
    D Nmapln               77     80B 0
    D Trgreserv3           81     96
    D*
    D* Pointers for record reference
    D  Intarr       S           1A   based(intptr) dim(32767)
    D  Intptr        S             *
    D*
    D Parm2          ds
    D Len                        9b 0

    D Bimage         S             *
    D Aimage         S             *

    D Bfile        E DS                EXTNAME(BEDISC)
    D                                  BASED(Bimage)
    D                                  PREFIX(B)
    D Afile        E DS                EXTNAME(BEDISC)
    D                                  BASED(Aimage)
    D                                  PREFIX(A)
    D*
    D UpdNetc        PR                EXTPGM('QCMDEXC')
    D   Cmd                     3000A   OPTIONS(*VARSIZE) CONST
    D   Cmdlen                     15P 5 CONST
    D*
    DCmdStr          S          1000    VARYING
```

Figure 383. Discount Table Trigger Program Source Code (Part 1 of 2)

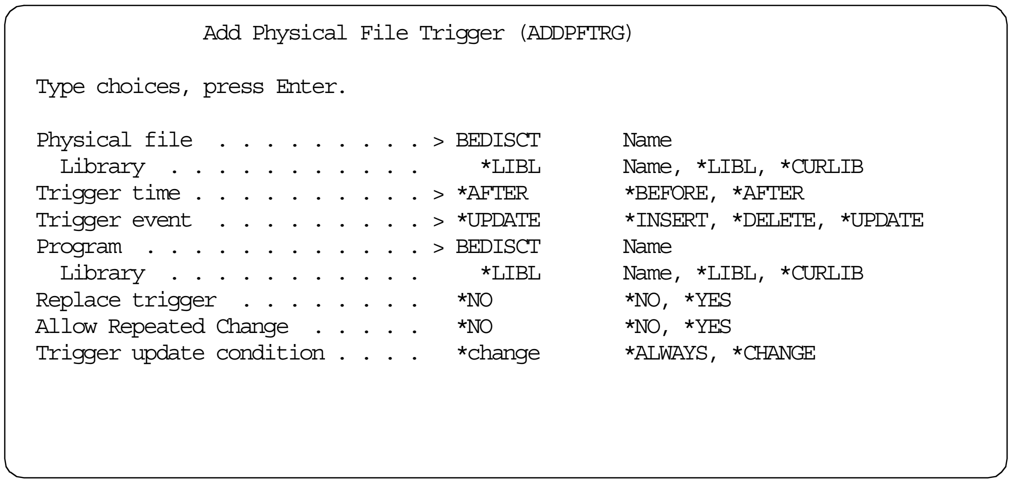
```
      D*****************************************************************
     D*                   ***   MAIN   ***                        *
    D*****************************************************************
     C
  001 C                 EXSR     Init
     C                 EXSR     ClrCach
     C
     C                 RETURN
    C*************************************************************
     C    ClrCach      begsr
    C**************************************************************
     C*
     C* In case of discounts policy change we must clear
     C* Net.Commerce cach
     C* Our shop uses only 1 merchant number
     C*
     C                EVAL     CmdStr = 'CLRCACH MERCHANT(28)' +
     C                                  ' PRODUCT(' +
     C                                 %trim(ABDPNBR) + ')'
     C                 CALLP    UpdNetc(CmdStr:
     C                               %len (CmdStr))
     C*
     C                 ENDSR
    C*************************************************************
     C    init         begsr
    C**************************************************************
     C    *entry       plist
     C                 parm                  parm1
     C                 parm                  parm2

     C                 eval      intptr = %addr(parm1)
     C                eval     bimage = %addr(intarr(oldofs+1))
     C                eval     aimage = %addr(intarr(newofs+1))



     C                 endsr
```

Figure 384. Discount Table Trigger Program Source Code (Part 2 of 2)

To activate the process, we must attach the trigger program to the BEDISCT table using the ADDPFTRG for three events:

- Insert
- Update
- Delete

Figure 385 on page 398 shows an example of attaching the trigger program for the update event. Since we set the trigger update condition parameter to *CHANGE, the program will be called only if the row really changed. That means that an update operation with equal before and after images of the row will not fire the trigger.



```
             Add Physical File Trigger (ADDPFTRG)

Type choices, press Enter.

Physical file  . . . . . . . . . > BEDISCT       Name
  Library  . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
Trigger time . . . . . . . . . . > *AFTER        *BEFORE, *AFTER
Trigger event  . . . . . . . . . > *UPDATE       *INSERT, *DELETE, *UPDATE
Program  . . . . . . . . . . . . > BEDISCT       Name
  Library  . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
Replace trigger  . . . . . . . .   *NO           *NO, *YES
Allow Repeated Change  . . . . .   *NO           *NO, *YES
Trigger update condition . . . .   *change       *ALWAYS, *CHANGE
```

Figure 385. Attach the Trigger Program to the Discounts Table

17.4 Requesting Capture upon Order Fulfillment

Our example uses the back-end system to perform the order fulfillment. Shipment details are not known to Net.Commerce. Therefore, we must request a capture from our acquirer only when the order is fulfilled. When the Net.Commerce payment module was configured, we chose to perform automatic approval and manual capture by the payment server.

Once the order is fulfilled, the back-end system should signal Net.Commerce that it can now capture the order payment. Figure 386 on page 399 displays the way our back-end system interacts with the Net.Commerce payment module.

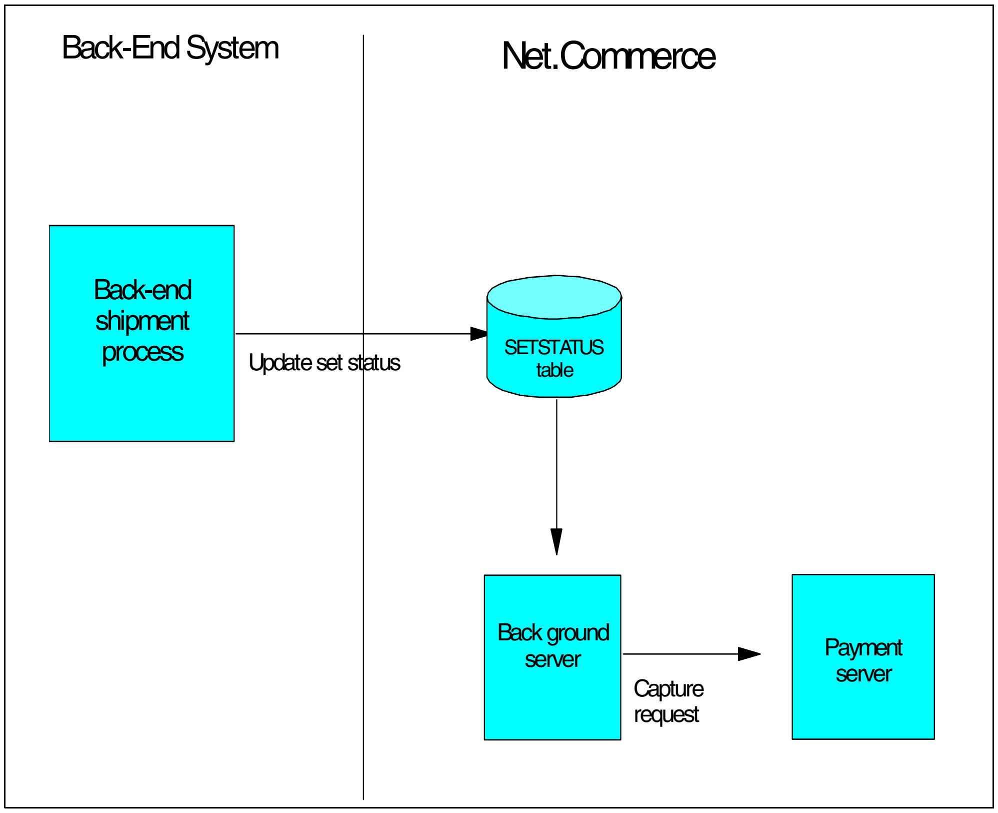


Figure 386. Order Payment Capture Request Mechanism

The back-end system order fulfillment process fulfills the order. It needs to update Net.Commerce table SETSTATUS and set the order status from "Capture ready" to "Capture required". The update can be performed from the Net.Commerce administration forms or automatically by the back-end system.

To perform the update automatically, the back-end system should issue the following SQL statement:

```
UPDATE SETSTATUS
SET SETSSTATCODE = 7
WHERE setsornbr = MyOrderNumber and
setsmenbr = MyMerchantNumber
and setsstatcode = 21
and setsauthamt = Order amount
```

In this SQL statement, note these points:

- `MyOrderNumber` is the Net.Commerce order number.
- `MyMerchantNumber` is the Net.Commerce merchant number.
- `setsstatcode= 21` is for an order with "Capture ready status."
- Order amount is the order amount that should be equal in the back-end system and Net.Commerce table.

This SQL statement can be issued by a back-end batch process or even as the ending stage in the back-end order fulfillment process. To manually change the order status from "Capture ready" to "Capture required," refer to the document *The Net.Commerce Payment Module*.

Appendix A, “Source Code Samples” on page 463, contains a sample RQSCAP command that can be used by the back-end system to change the order status and trigger capture request.

Net.Commerce includes a background server that periodically processes background requests, including requests for capture, capture reversal, credit, and credit reversal. The server sends the request to the payment server. Then, the Net.Commerce server receives the responses from the payment server through user exits and changes the state of the affected transactions accordingly. The payment module communicates with the payment server using sockets. The payment module also supports merchant-originated purchased by using the pay_accept command. The change in the SETSTATUS table is noted by the Net.Commerce background process. It issues a capture request to the payment server that will send the request to the acquirer.

Note

Our example is very simple. You may have to perform a validity check before doing a direct update of the order status.

17.5 Usage Considerations

The following list points out some issues that we needed to consider for our example. Please note that this is only a partial list, and you may find many more issues to consider on your own Net.Commerce-to-back-end integration solution.

- Net.Commerce overridable functions refer to objects that are located in the back-end production library. We must include the back-end production library in Net.Commerce jobs library list. The startup program listed takes care of it.
- Some back-end functions refer to Net.Commerce tables. For example, the CLRCACH utility is called from the back-end product table trigger program. It reads and writes data to Net.Commerce tables. We must make sure that the Net.Commerce table will be available to the CLRCACH utility in our example. We compiled the CLRCACH command and set the current library parameter to the Net.Commerce schema as shown here:

  ```
  CRTCMD CMD(CLRCACH) PGM(CLRCACH) CURLIB(WORK)
  ```

- To make sure the back-end integration has consistent behavior, we must make sure that all processes start in the same time. The start-up program shown in Figure 387 on page 401 does that.

```
/*************************************************************/
 /* This program will start Net.Commerce and take care of back  */
 /* end integration issues.                                   */
 /*                                                           */
 /*************************************************************/
             PGM        PARM(&pr_inst)

             DCL        VAR(&pr_inst)    TYPE(*CHAR) LEN(10)

             DCL        VAR(&libl   )    TYPE(*CHAR) LEN(275)
             DCL        VAR(&cmd    )    TYPE(*CHAR) LEN(512)

             RTVJOBA    USRLIBL(&libl)

 /* Setting the back end production library to be included */
            ADDLIBLE   LIB(NETCBE ) POSITION(*LAST)
            MONMSG     MSGID(CPF2103) /* Already in libray list */
            STRNETCSVR INSTANCE(&pr_inst)

/* Starting our resdint program to bridge Net.Commerce order */
/* request to the back end system. We will add the instance  */
/* library here also. */

            ADDLIBLE   LIB(&pr_inst) POSITION(*LAST)
            MONMSG     MSGID(CPF2103) /* Already in libray list */
            SBMJOB     CMD(CALL PGM(EXTORDERR)) JOB(NETCGW)

/* Start the Net.Commerce cache daemon to make sure outdated */
/* pages are purged from the cache.                          */

            QNETCOMM/STRNETCDMN INSTANCE(&pr_inst)

 /* Return the original library list */
             CHGVAR     VAR(&cmd) VALUE('chglibl libl(' *TCAT &libl +
                          *TCAT ')')
             CALL       PGM(QCMDEXC) PARM(&cmd 512)

             ENDPGM
```

Figure 387. Net.Commerce Start Up Program STRNETCBE

- There are several ways to integrate e-mail confirmation. In our example, we used the SNDEMAIL command that was published in the September 1998 issue of *NEWS/400* magazine in the article entitled "RPG Utility Puts QtmmSEndMail API to Work". You can also use the `SNDDST` command or directly call the QtmmSendMail API.

Note

Net.Data now supports sending e-mail using the DTW_SENDMAIL function. This function is included as part of PTF SF57236 for licensed program product 5769-DG1. For more information about this function, refer to the Web site at: `http://www.as400.ibm.com/netdata`

- In our shop, we did not use shopper registration. However, if your back-end system has existing customers, you may find it convenient to load the data from your customer back-end table to the Net.Commerce shopper table. One issue to consider when importing customers data is the fact that Net.Commerce uses an encrypted password. That is, if your back-end system has password information in its tables, you cannot directly store the password information in Net.Commerce tables. Section A.1, “Retrieving Encrypted Text” on page 463, provides an example of a utility that can be used to encrypt the password taken from your back-end customer table before writing it in the Net.Commerce shopper table.

17.6 Relevant Tables and Programs

For the completeness of our solution description, the following tables are the relevant database tables and programs that we use to develop Net.Commerce to back-end system integration.

Table 21. Back-End System Relevant Tables

Table name	Description
BEPROD	Products table
BECATEG	Product category table
BEDISC	Discount table
BEWORK	Orders work file (The back-end system uses this file to process orders)
BEMEASUR	Measurement description table

The back-end table layout is described in A.6, “Back-End Table Definition” on page 472.

Table 22. Programs and OFs Used to Interact with the Back-End System

Program name	Description	Language	Source Location
GETPRICER	Gets products price from the back-end system.	RPG	Chapter 19, “Implementing Overridable Functions” on page 413
BEDISCT	Trigger program for the discount table.	RPG	Figure 384 on page 397
BEPRODT	The trigger program for the products table.	RPG	Figure 318 on page 341
CLRCACH	Signals the expiration of a cached Web page.	RPG	Appendix A, “Source Code Samples” on page 463
EXTORDERR	Retrieves new order from Net.Commerce and creates an order in the back-end system	RPG	Figure 381 on page 391
GETPRICE	Overridable function called by Net.Commerce to retrieve price.	C++	Figure 413 on page 429
EXTORDER	Overridable function called by Net.Commerce to signal a new order.	C++	Figure 378 on page 387
ORDERC	Sends e-mail confirmation.	CL	Appendix A, “Source Code Samples” on page 463

Chapter 18. Generating Net.Commerce Reports

Net.Commerce lets you quickly set up and populate a mall from which shoppers can browse and order. The site activity is written to the Net.Commerce database. The Net.Commerce database schema is fully described in the book *Net.Commerce Commands, Tasks, Overridable Functions, and Database Tables*.

You can query the data in the Net.Commerce database to gather important statistics information about orders and sales. To query the database, you must be familiar with the Net.Commerce database schema. You must also know the names and contents of relevant tables and columns in the database.

This chapter briefly describes some of the possibilities for producing reports based on the Net.Commerce database schema.

18.1 Integrating Seagate Crystal Report 6 with IBM Net.Commerce

The Net.Commerce Web site contains a downloadable tool that allows you to gain access to your data through Seagate Crystal Reports. A set of 17 Net.Commerce-specific sample reports are available to help you start with Net.Commerce reporting. You can download these reports and change or extend them to fit your particular reporting and analysis needs. The reports include sales reports and various summary statistics. To test and run these reports, you should have a registered version of Seagate Crystal Reports version 6 and a working ODBC connection to the Net.Commerce AS/400 server.

The reports with their installation and configuration manuals are available on the Web at: `http://www.software.ibm.com/commerce/net.commerce/isv/download.html`

You can also use the reports to learn more about the Net.Commerce schema and database relationship, so that you can create your own SQL request to the Net.Commerce database. The collection of reports included with this package is not intended to be an exhaustive list of all reports that your business may need or all reports that can be generated from the Net.Commerce database.

Currently the following reports are available in this tool:

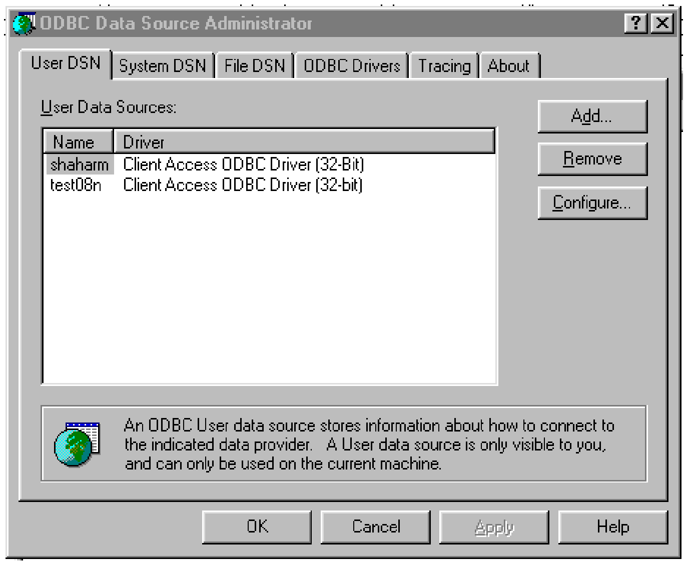
- Crystal reports Net.Commerce reports
- Interest Item Statistics
- Pending Order Statistics
- Order Status Statistics
- 5 Largest Pending Orders
- 5 Largest Completed Orders
- Sales by Product
- Sales by Product Category
- Sales by Merchant
- Sales by Merchant Category
- Sales by Shopper Group
- Sales by Time of Day
- Sales by Day of Week
- Sales by Geography

18.1.1 ODBC Driver Configuration

Crystal Reports connect directly to the Net.Commerce database using an ODBC connection. We used the *IBM Client Access ODBC driver* to run the reports.

To configure an ODBC connection to the AS/400 system from the Windows 95 client using the IBM Client Access ODBC driver, complete these steps:

1. Click the **Start** button.
2. Follow this path: **Setting —> Control Panel —> 32BIT ODBC**
3. Click on the **System DSN** tab.



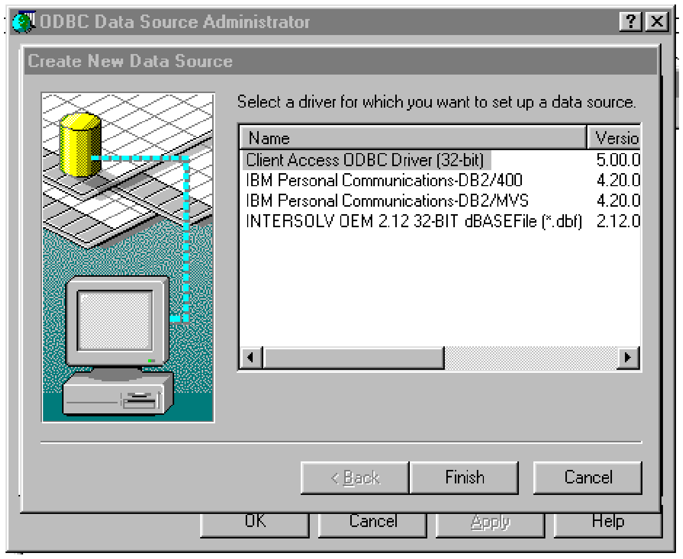
User Data Sources:

Name	Driver
shaharm	Client Access ODBC Driver (32-Bit)
test08n	Client Access ODBC Driver (32-bit)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Figure 388. Configure ODBC — The ODBC Administrator

4. Click on the **Add** button. The Create Data Source window appears (Figure 389).

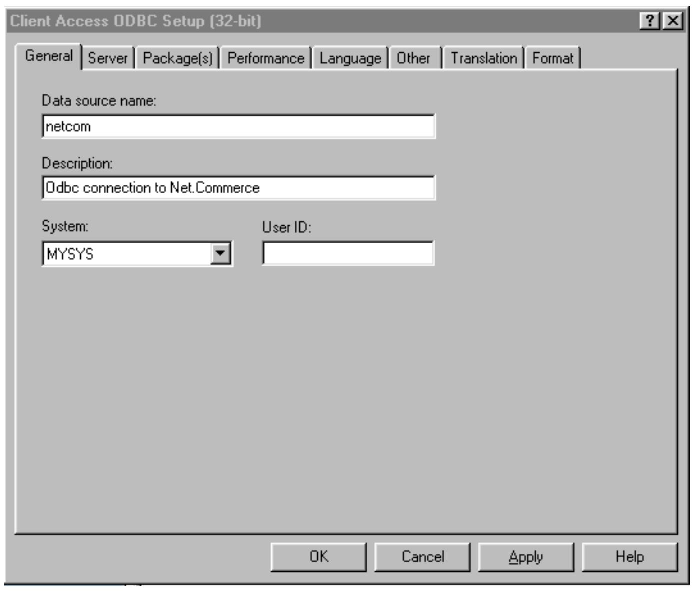


Select a driver for which you want to set up a data source.

Name	Versio
Client Access ODBC Driver (32-bit)	5.00.0
IBM Personal Communications-DB2/400	4.20.0
IBM Personal Communications-DB2/MVS	4.20.0
INTERSOLV OEM 2.12 32-BIT dBASEFile (*.dbf)	2.12.0

Figure 389. Configure ODBC — Create a New Data Source

5. Click on the **Finish** button. The client access ODBC setup appears.



Data source name: netcom

Description: Odbc connection to Net.Commerce

System: MYSYS

User ID:

Figure 390. Configure ODBC — Client Access Driver Set up

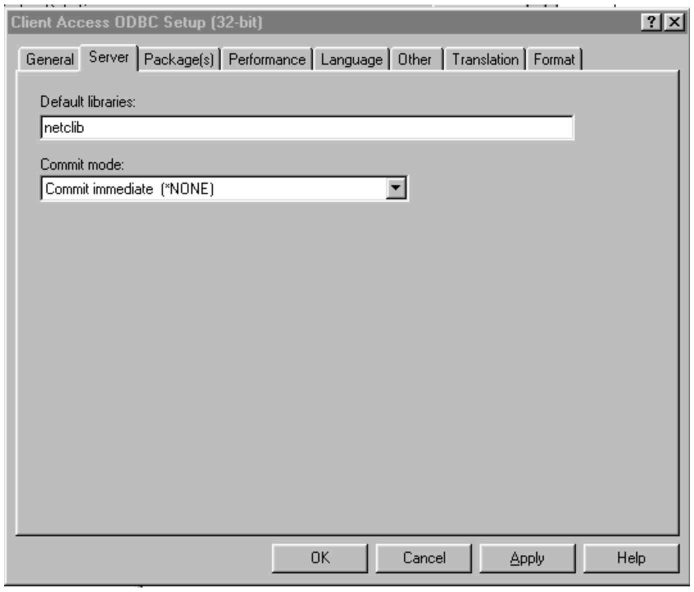
6. Fill in the following parameters:

   In the *Data source name,* write a meaningful name for your data source. You use this name in the Crystal Report set up.

   In the *description,* write a short description to your data source.

   In the *System,* specify the configured AS/400 system that contains the data source. Click on the down arrow to select another system.

7. Click on the **Server** tab.



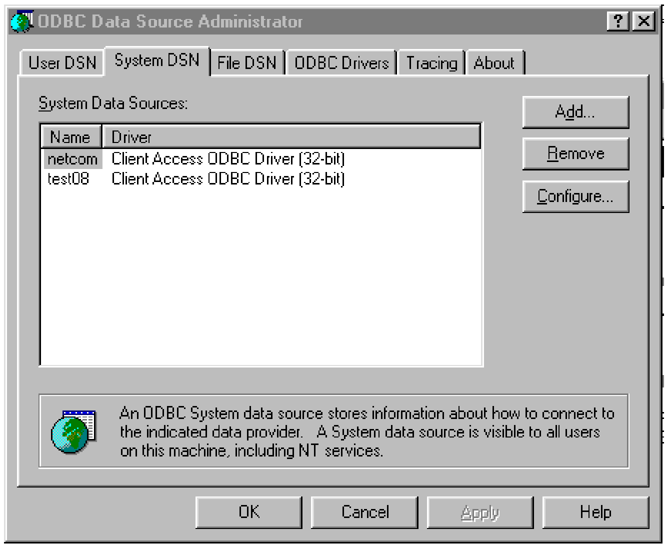
Default libraries: netclib

Commit mode: Commit immediate (*NONE)

Figure 391. Configure ODBC — Specify the Net.Commerce Library

8. In the default library, specify the Net.Commerce schema library name. This will be the name of the user profile who owns the schema.

9. Click **OK**



System Data Sources:

Name	Driver
netcom	Client Access ODBC Driver (32-bit)
test08	Client Access ODBC Driver (32-bit)

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

Figure 392. ODBC — End of Driver Setup

Your data source name should now appear on the screen. Any ODBC-compliant application, including Crystal Reports, can connect to the Net.Commerce schema using the data source name you chose.

To run the Crystal Reports, follow the instructions described in the manual *Integrating Seagate Crystal Reports v.6 with IBM Net.Commerce.* This manual was downloaded from the Web with the reports set.

18.2 Creating a User-Defined Reports Example

Let us assume, for example, that the mall administrator wants to receive a daily report that shows them the total sales for that day. To create the report, we must complete these steps:

1. Identify the relevant tables in the Net.Commerce database.
2. Identify the columns and the selection criteria.
3. Issue the SQL request for the required report. If the request will be sent from an ODBC compliant application, we can use this application for report generating.

18.2.1 Identifying the Relevant Net.Commerce Tables and Columns

The document *Net.Commerce commands, Tasks, Overridable Function, and Database Tables* is the best source to find information regarding tables and columns in the Net.Commerce database schema. In the previously mentioned document, we see the following description for the ORDERS table:

> *"The ORDERS table contains information about orders that are placed by shoppers. Each row corresponds to a single order. The products and items in an order may have different shipping information. That is, the products and items can be shipped to different locations."*

Since we will not use shipping information in this report, it seems that the ORDERS table will be suited for our report. The ORDERS table also contains a merchant reference number that is a foreign key to the merchants table. Therefore, we may use the MERCHANT table to retrieve the merchant name.

The ORDERS table description contains a detailed explanation for each column. The following columns are important for our example report request:

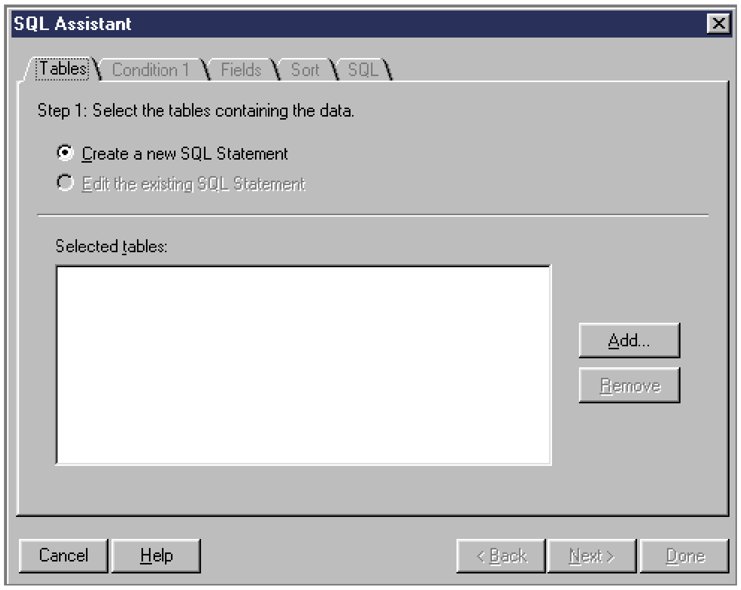
- **ORMENBR** — Merchant reference number that will be used to retrieve the merchant name from the MERCHANT table.
- **ORPRTOT** — Total product price for the order. We will assume that all sales are in the same currency.
- **ORTXTOT** — Total sales tax for this order.
- **ORSTAT** — Order status. We will exclude pending ("P") and cancelled ("X") orders from our report.
- **ORPSTMP** — The date and time that the order was placed. We will use this column to select only the orders placed today.

18.2.2 Creating the Example Report Using Lotus Approach

We use Lotus Approach to connect to the Net.Commerce database with ODBC and produce our example report. It is important to understand that the same SQL request we used in this example can be issued directly on the server (using SQLUTIL for example). It can also be issued from another PC-ODBC-compliant application such as a Seagate Crystal Report.

To produce the report, follow these steps:

1. Define an ODBC connection to the AS/400 server where the Net.Commerce database is located as described in the previous section.
2. Launch Lotus Approach. Click **Cancel** on the Welcome page.
3. Click **File —> Open edit SQL**. The SQL Assistant window appears (Figure 393 on page 408).



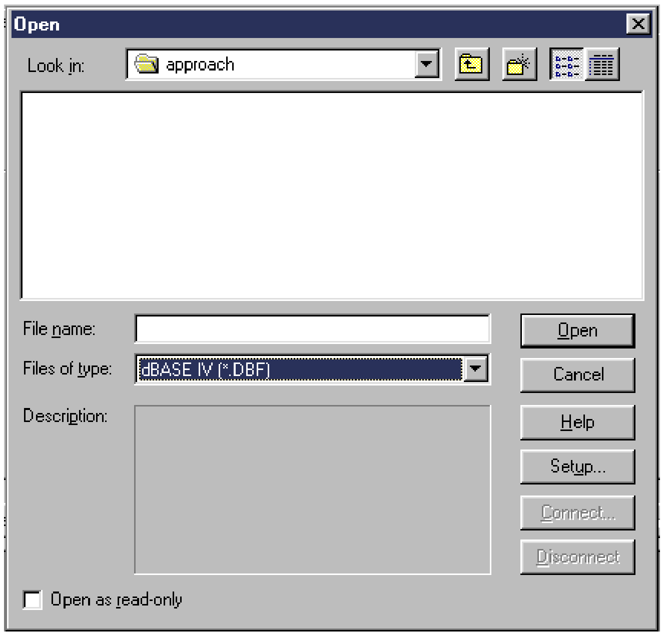
Step 1: Select the tables containing the data.

(●) Create a new SQL Statement
( ) Edit the existing SQL Statement

Selected tables:

Figure 393. Report Definition — Tables Selection

4. Click on the **Add** button. The Open dialog is displayed (Figure 394).



Look in: approach

File name:

Files of type: dBASE IV (*.DBF)

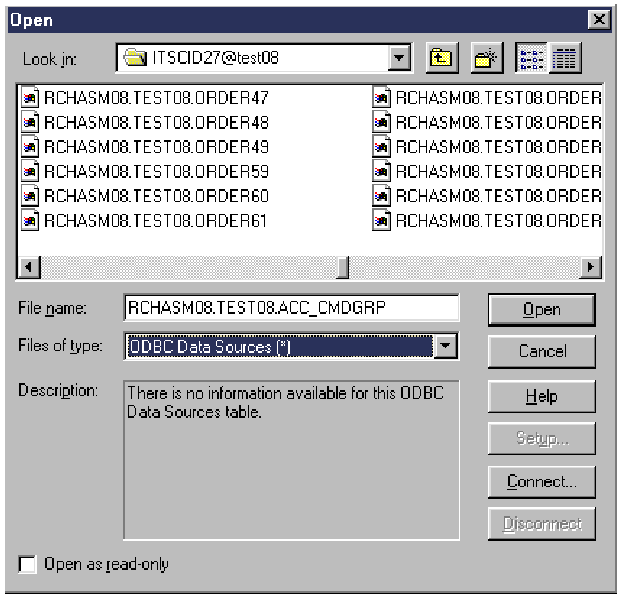
Description:

☐ Open as read-only

Figure 394. Report Definition — Select Database Type

5. Select files of the type ODBC Data Sources. You will see a list of available data sources. Choose the data source name you defined earlier. Then, click the **Open** button.

   A list of available tables on the Net.Commerce database is displayed (Figure 395 on page 409).


Open

Look in: ITSCID27@test08

RCHASM08.TEST08.ORDER47
RCHASM08.TEST08.ORDER48
RCHASM08.TEST08.ORDER49
RCHASM08.TEST08.ORDER59
RCHASM08.TEST08.ORDER60
RCHASM08.TEST08.ORDER61
RCHASM08.TEST08.ORDER
RCHASM08.TEST08.ORDER
RCHASM08.TEST08.ORDER
RCHASM08.TEST08.ORDER
RCHASM08.TEST08.ORDER
RCHASM08.TEST08.ORDER

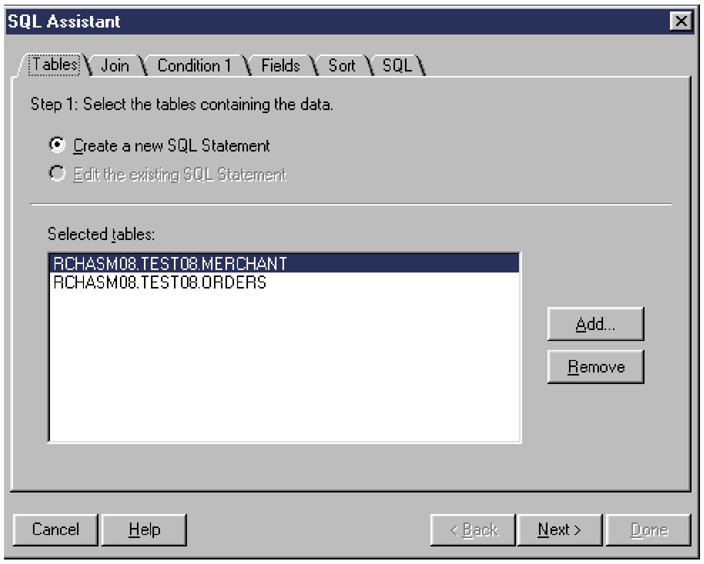
File name: RCHASM08.TEST08.ACC_CMDGRP
Files of type: ODBC Data Sources (*)
Description: There is no information available for this ODBC Data Sources table.

Open as read-only

Figure 395. Report Definition — List of Available Tables

6. Select the **MERCHANTS** table and click the **Open** button. Click the **Add** button. Select the **ORDERS** table and click **Open** again.

   You should now see the SQL assistant screen with the two table names.


SQL Assistant

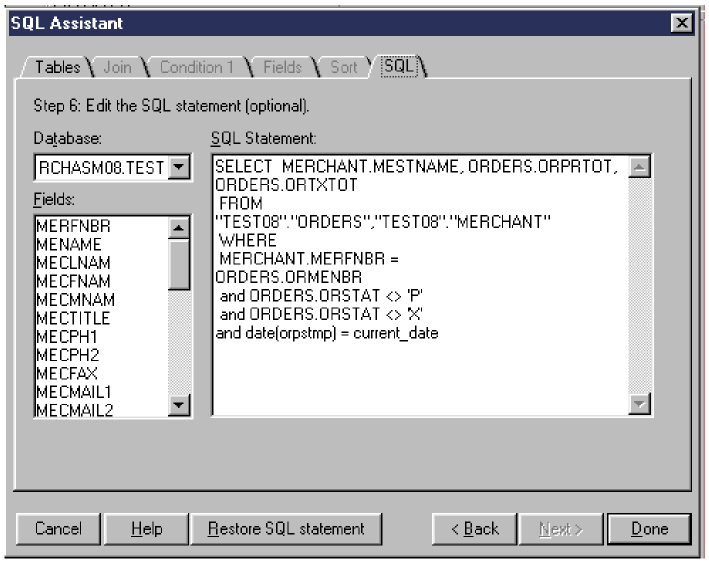
Step 1: Select the tables containing the data.

(●) Create a new SQL Statement
( ) Edit the existing SQL Statement

Selected tables:
RCHASM08.TEST08.MERCHANT
RCHASM08.TEST08.ORDERS

Figure 396. Report Definition — Selected Tables

7. Click on the **SQL** tab to directly code the SQL statement (Figure 397 on page 410).



Step 6: Edit the SQL statement (optional).

Database: RCHASM08.TEST

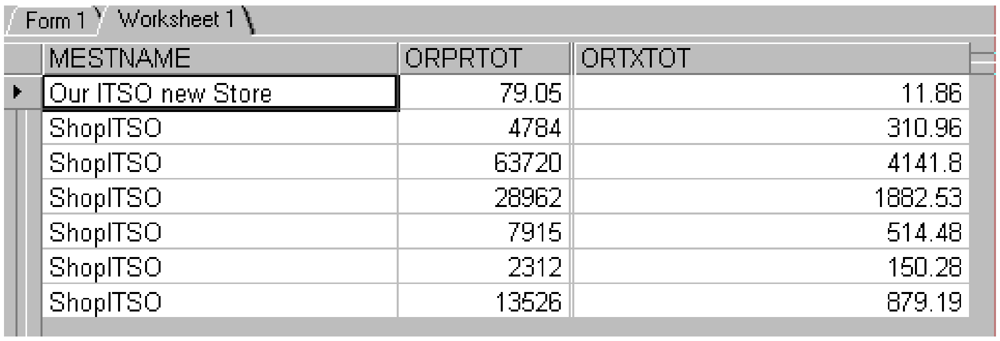
Fields: MERFNBR, MENAME, MECLNAM, MECFNAM, MECMNAM, MECTITLE, MECPH1, MECPH2, MECFAX, MECMAIL1, MECMAIL2

SQL Statement:

```
SELECT  MERCHANT.MESTNAME, ORDERS.ORPRTOT,
ORDERS.ORTXTOT
 FROM
"TEST08"."ORDERS","TEST08"."MERCHANT"
 WHERE
 MERCHANT.MERFNBR =
ORDERS.ORMENBR
 and ORDERS.ORSTAT <> 'P'
 and ORDERS.ORSTAT <> 'X'
and date(orpstmp) = current_date
```

Figure 397. Report Definition — SQL Statement

8. Click on the **Done** button. You receive the report results as shown on Figure 398.



MESTNAME	ORPRTOT	ORTXTOT
Our ITSO new Store	79.05	11.86
ShopITSO	4784	310.96
ShopITSO	63720	4141.8
ShopITSO	28962	1882.53
ShopITSO	7915	514.48
ShopITSO	2312	150.28
ShopITSO	13526	879.19

Figure 398. Define Reports — SQL Statement Results

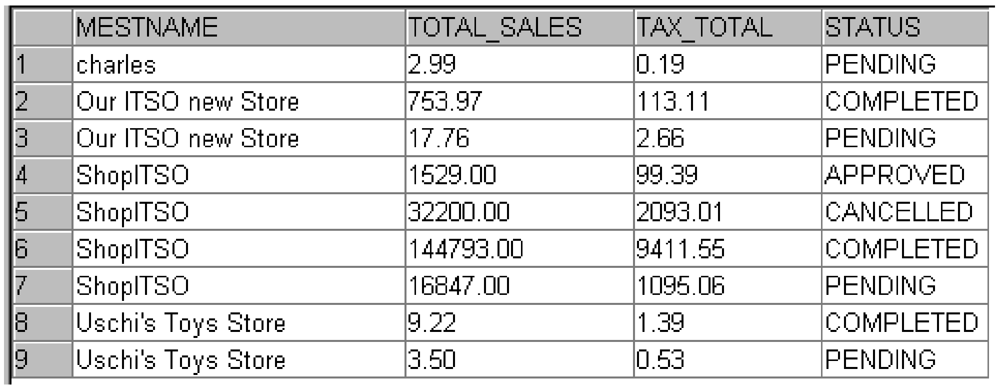
Now that the SQL results are in the Lotus Approach worksheet, you can use the Lotus Approach product to generate reports with grouping, summary, different sort columns, and so on.

Let us consider a second example. The store manager wants to see totals from orders in the last 30 days that are grouped by the merchant and invoice type. We will use the MERCHANT and ORDERS tables to produce the report. The SQL statement that we issue looks like this example:

```
SELECT  MESTNAME, SUM(ORPRTOT) AS TOTAL_SALES,
  SUM(ORTXTOT) AS TAX_TOTAL,
case orstat
when 'P' then 'PENDING'
when 'C' then 'COMPLETED'
when 'X' then 'CANCELLED'
else 'APPROVED'  END    AS STATUS

FROM ORDERS , MERCHANT

WHERE
ORMENBR = MERFNBR
AND DATE(ORUSTMP ) > CURRENT_DATE - 30 DAYS
```

```
GROUP BY
MESTNAME, ORSTAT

ORDER BY MESTNAME,STATUS
```

This SQL statement returns the results shown in Figure 399.



	MESTNAME	TOTAL_SALES	TAX_TOTAL	STATUS
1	charles	2.99	0.19	PENDING
2	Our ITSO new Store	753.97	113.11	COMPLETED
3	Our ITSO new Store	17.76	2.66	PENDING
4	ShopITSO	1529.00	99.39	APPROVED
5	ShopITSO	32200.00	2093.01	CANCELLED
6	ShopITSO	144793.00	9411.55	COMPLETED
7	ShopITSO	16847.00	1095.06	PENDING
8	Uschi's Toys Store	9.22	1.39	COMPLETED
9	Uschi's Toys Store	3.50	0.53	PENDING

Figure 399. Orders by Merchant and Status Example Report

You can now use Lotus Approach capabilities to manipulate and produce reports using the result set of the SQL request.

Of course, these examples are very simple. For more complicated reports, you may need to make a greater effort in finding the correct tables, columns, and SQL statements that are required for the report.

You may find that the data stored in the Net.Commerce database schema can be very useful to retrieve important statistics about your site. We recommend that you take the time to become familiar with the Net.Commerce database schema so that you can achieve the most from it.

Chapter 19. Implementing Overridable Functions

Net.Commerce is a front-end application that provides the tools to quickly set up a mall from which shoppers can browse and order. Net.Commerce uses API functions to implement the shopping process. The ideal situation is to map the entire site flow directly to existing Net.Commerce API. However, if Net.Commerce meets only part of our requirements, we must tailor and extend the system by using the Net.Commerce commands and overridable functions.

This chapter introduces the basic model of the Net.Commerce API and describes the steps needed to implement user-written overridable functions.

Note

Net.Commerce documentation refers to AS/400 service programs as DLLs.

19.1 The Basic Model

Commands and overridable functions represent the basic building blocks of the Net.Commerce system. They are the components that allow you to extend the base system. Typically, you have commands that call up tasks, which will be mapped to overridable functions. Commands indirectly call overridable functions through tasks. They can also indirectly call other commands, and overridable functions can call other tasks. Then, these components access the database.

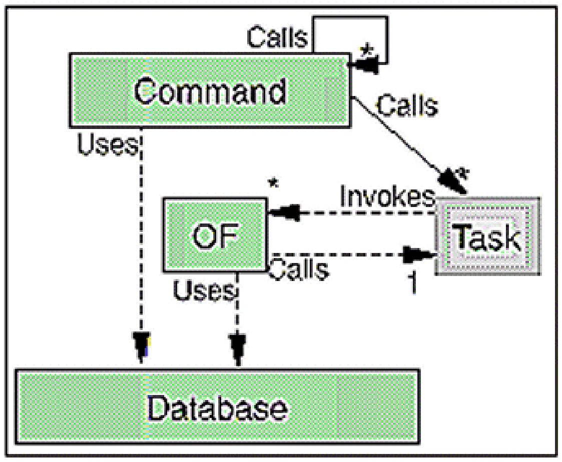


Figure 400. Net.Commerce Basic Model

At a high level, the idea is that a command represents a static business process that delegates well-defined pieces of business logic to tasks. Through system configuration, the administrator can map a given implementation for the business logic to a task for a merchant. The same command is hard coded to call one or more tasks. Depending on the merchant involved (a dynamic property based on various input to the command), possibly different overridable functions will be called. For example, let us consider a simplified version of the OrderItemProcess command. This command takes in a list of SKUs mapped to quantities, and adds them to orders. This command can accept a list of products from different merchants. As part of adding a product to an order, we need to check the current inventory for the product and capture its price. This command’s generic algorithm can be represented as shown here:

```
For every {sku=quantity} name/value pair in the HTTP request
{
CheckInventory(Sku, Quantity);
Price = GetPrice(Sku);
Add the product, with its price, and the quantity requested in an order
}
```

The designer of this command decided that it was not realistic to generically check the inventory on a product, or compute its price. Therefore, tasks were created so that a third party could implement these operations as overridable functions. Also, the command can accept an arbitrary list of products and the command can be executed in a mall context where several merchants cooperate. As a result, several implementations can actually be invoked in the loop. The task mechanism allows the separation of concern between the command writer (whose job it is to make the command as generic as possible while still defining a business process) and the overridable function writer (whose job is to implement a specific piece of business logic for a merchant). Tasks also allow the ability to transparently invoke different implementations for the same operation based on merchant parameters.

The *OF manager* is the code that actually physically invokes an overridable function. The command only specifies the task to be run, its parameters, and the merchant for which it should be run. To be closer (but not exact) to the actual implementation of a command, the algorithm would appear as shown here:

```
For every {sku=quantity} name/value pair in the HTTP request
{
Determine merchant for sku;
OF_Manager.Call(“CHK_INV”, Merchant, Sku, Quantity);
Price = OF_Manager.Call(“GET_BASE_UNIT_PRC”, Merchant, Sku, Quantity);
Add the product, with its price, and the quantity requested in an order
}
```

Here, each merchant involved can choose to either implement their own version for each task or use the default versions that Net.Commerce ships.

Commands and overridable functions are actual pieces of code, but tasks are not. Tasks are more like specifications. They are contracts that define the behavior and the input and output parameters that a calling command and a called overridable function must abide by for the system to function properly. Tasks are really names for commands and overridable functions to communicate with one another. Tasks are virtual objects that you would document as shown here:

```
Task : CHK_INV
Semantics: Given a SKU S and a Quantity Q, checks whether there is enough
inventory for the user to order Q of S.
Input : SKU reference number “SKU_REF_NUM” (integer value in a String)
Quantity “QUANTITY” (floating point value in a String)
Output : None
```

In summary, commands and overridable functions are the components that make up the runtime of a Net.Commerce server. They represent the business process and logic involved in running an e-commerce site respectively. Commands invoke overridable functions indirectly through tasks that define the interfaces. Overridable functions are implemented separately from commands, and are “plugged in” by a Net.Commerce administrator. Commands and overridable functions share the same programming model, and are managed similarly at the system level, but they fill a different purpose:

- Commands are entry points into the system. They correspond to a URL. Overridable functions are used internally and are accessed indirectly through tasks.
- Commands represent a business process (for example, process an order). Overridable functions represent precise pieces of business logic (for example, update product inventory).
- Overridable functions exist at the merchant level. Each merchant in a mall can provide its own overridable functions for a given Task. Commands exist at the system level. This does not mean that a merchant cannot replace a command. It means that one URL cannot be mapped to multiple commands with the system resolving which one to invoke based on the merchant.
- Commands have a complex runtime model. Overridable functions are lightweight. They are basically function objects.
- Commands have access control.
- Mathematically speaking, commands are complex because they deal with a larger model of the world (many commerce objects, other commands, and tasks). Overridable functions are simple, meaning that their task is very well defined. Paradoxically, commands are simpler to implement because they work at a higher level. Overridable functions can be daunting to implement because they deal with issues at a much more detailed level.

For example, we can look at the OrderProcess command described in Figure 401. The command invokes tasks such as UpdateInventory, DoPayment, and so on. Later on, an actual implementation is provided for each task: an overridable function.

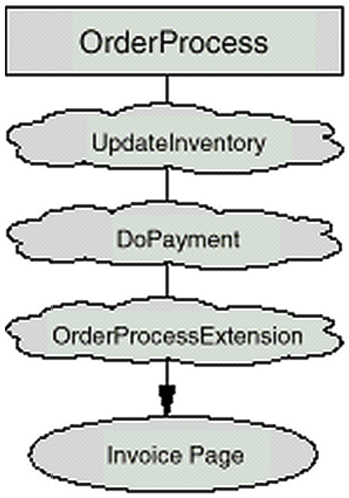


Figure 401. Flow of the OrderProcess Command

In our example, the command contain three tasks. Each task defines an interface between the command and the overridable function. They will be implemented by Net.Commerce default overridable functions or by merchant-written overridable functions. For example, if your shop uses a back-end application and inventory should be updated on the back-end tables, you can create your own overridable functions that implement the UpdateInventory task.

Commands are invoked by a URL request from the shopper browser. For example, the following example shows product number 2 for merchant number 12345:

```
http://host_name/cgi-bin/ncommerce/ProductDisplay?prmenbr=12345&prrfnbr=2
```

19.2 Tasks and Overridable Functions

There are two types of tasks. Some tasks are meant to do processing-type work, while others are meant to render or generate a page to be sent back to the browser. Of these display tasks, *view tasks* and *error tasks* basically correspond to a success or failure state. *Display tasks* are generally the last step that occurs in a command. There are other types of tasks, but we do not expose them. System tasks are important to know about since they are invoked, and they show in the logs. However, they are reserved for IBM use and are not customizable by end-users.

Overridable functions are implementations associated with tasks. An overridable function implements the semantics and interface of a task so that a command can call it. There is a direct relationship between an overridable function and the task for which it was written. For an overridable function to be compatible with a task, it must obviously implement a behavior that is compatible with that specified by the task.

Overridable functions associated with display tasks, whether error or view, are responsible for populating the HTTP request with a browser-viewable document. The Net.Commerce system has implemented two overridable functions that use Net.Data to render pages. One of these overridable functions takes in a file to render, while the other uses its invoking task and information in the database to get a filename back. Because Net.Data is the runtime renderer that Net.Commerce uses, these are the only overridable functions that Net.Commerce provides for all display tasks. However, you can code a view directly in your own overridable function.

All other overridable functions are associated to process tasks. These functions are meant to perform some computational work such as checking inventory for a product, computing a discount price, or invoking a payment sub-system. In case of an error, most process overridable functions invoke an error task. In general, overridable functions can invoke tasks the same way commands would invoke them. Obviously, an overridable function should not invoke the task under which it itself was invoked. Otherwise, an infinite loop would ensue. Also, although there is nothing that would prevent an overridable functions from calling a command, such a situation would render the runtime model quite complex. Therefore, it is not recommended.

19.3 General Issues

Now that you understand the basic architecture of Net.Commerce, it is time to consider some general Net.Commerce issues before you begin the actual coding.

19.3.1 Command-Oriented Programming

When creating applications for the Internet, you are actually programming in a client/server environment. The server is there to execute the commands that are invoked by the client. After a command is finished, data is presented to the end user, who usually sits in front of an Internet browser such as Netscape Navigator or Microsoft Internet Explorer. This is in contrast to writing a program for a stand-alone computer. Here, you only have to think about one user, and you can always make requests to the user about a file name or any other input that you need at that specific moment.

The programing method used in a client/server environment is more like sending command requests to a server that performs all the execution logic and then comes back with a response to the end user. This is much like the old-fashioned text adventure games where you could type commands such as “go north,” “look,” and “open door.” After such a command was processed, a new message was displayed to the end user saying, for example, “the door is already open” or “you need a key to open the door.”

At this stage, we already see a pattern emerging. We need to have a way of getting command requests from an end user, executing program logic (the commands and overridable functions), and displaying data to the end user.

When implementing an e-commerce solution, the processing logic is never simple. It can involve such processes as performing advanced calculations, working against the database, working side-by-side with systems like SAP, handling EDI documents, and managing MQ series queues. Also the internal logic of the store must be taken care of. We can in fact extend our similarity to adventure programs and create a store where you issue such commands as “put three eggs in the shopping cart,” “look at the bread shelf,” or “give the shopping cart to the cashier.” In fact, these commands have their Net.Commerce equivalents in the InterestItemAdd, CategoryDisplay, and OrderItemProcess commands. Instead of just presenting the results as plain text, Net.Commerce generates HTML pages that can be viewed by the end user.

19.3.2 Programming with C++

C++ is the programming language for Net.Commerce commands and overridable functions. C++ is an industry-standard programming language that has been used over and over in a multitude of large-scale projects, including complete operating systems. It has everything that is required for performing advanced systems programming. Do not be afraid of using C++ when programming in Net.Commerce, because you do not have to know every aspect of the language to use it.

Because the command execution is done on the server side, you never have to use class libraries for creating menus, windows, and other operating-system dependent functions. You quickly find that you are using some specific code patterns over and over again when you are doing command programming. You can also use C++ templates and call your own RPG programs from the C++

template. As mentioned before, you do not need to know everything about C++ to write Net.Commerce commands and overridable functions. However, if you are not familiar with such terms as class, object, instance, or method, you are encouraged to read a bit more about C++.

The AS/400 system supports C++ compiling in these two ways:

- Uses the VisualAge for C++ compiler (*5716CX4* for OS2 client, *5716CX5* for windows 95/NT client)
- Uses the AS/400 native C++ compiler PRPQ *5799-GDW*

19.3.3 Net.Commerce Classes

Net.Commerce includes a number of C++ classes used to program commands and overridable functions. These classes provide the framework for building the overridable functions and useful tools for performing common operations such as manipulating environment variables and accessing the database.

An overridable function may need to retrieve values from the HTTP request and use them during its processing. The *Web classes* provide methods for performing these functions. For example, the *HttpRequest* class is used to get or set HTTP request information. The *HttpResponse* class is used to set or get the HTTP document to be returned to the browser.

When a task is called from a command, parameters may be passed to the overridable functions using the *NC_Environment class*. When overridable functions return, it may be expected to pass a result in the NC_Environment. For example, the line of code is used to receive a value from the environment variables:

```
string * MerchantRefNum = (string *) Env.Seek(_PARAM_NAME_MERCHANT_REF_NUM);
```

You can access the Net.Commerce database from an overridable function. The *database classes* allow you to construct and execute SQL statements and retrieve the statements results. When you wish to access the database, you should acquire the current connection using the following statement:

```
DataBase* DB = (DataBase *) Env.seek(NC_Environment::_VAR_MAIN_DATABASE);
```

You can construct an SQL statement as a string and use the *SQL class* to execute it. You can use the *row class* to retrieve each row that is retrieved.

For a detailed explanation and description of each class and for coding guidelines, consult the guide *Commands, Tasks, Overridable Functions and the E-commerce Framework.*

19.3.4 Other Resources

Overridable functions allow you to extend and improve the functionality of Net.Commerce. However, their programming model is not trivial. To better understand the programming environment and classes, consider these resources:

- *Commands, Tasks, Overridable Functions and the E-commerce Framework*

  This document is available in softcopy format only from the Net.Commerce Web site. Refer to E.4, “Other Resources” on page 536, for information about obtaining this document.

- *Net.Commerce Commands, Tasks, Overridable Functions, and Database Tables*

  This document is available in the documents directory of your AS/400 after you install the Net.Commerce product. The directory path is: */QIBM/ProdData/NetCommerce/html/MRI2924/ncbooks*

- *Building an E-commerce Solution,* SG24-5417, Chapters 12 through 17.

19.4 Overridable Function by Example

Here are the steps for implementing your own overridable function:

1. Identify the parts of your site where the default behavior of Net.Commerce does not meet the requirements.
2. Define the new behavior that you need. Design the best solution to implement your needs.
3. Code and test a new overridable function that will implement the new behavior.
4. Replace the default overridable function with your own overridable functions using Net.Commerce administration forms.

19.4.1 Identifying the Need for New Overridable Functions

In the design phase of our site, we created a navigation flow diagram. Next, we tried to map the flow diagram to the Net.Commerce commands. We found out that we need to use the *OrderDisplay command*.

Looking in the documentation, we find that, among other things, the command *OrderDisplay* retrieves the product price by using the task GET_BASE_SPE_PRC. The default overridable function for the GET_BASE_SPE_PRC task is named *GetBaseSpePrc_1.0(IBM,NC).*

Here is a brief description of the *GetBaseSpePrc_1.0(IBM,NC)* overridable function:

- Retrieves the product and item prices from the PPPRC column in the PRODPRCS table for the appropriate shopper group and for the current date and time.
- Orders the result by the precedence level that is specified in the PPPRE column in the PRODPRCS table, and inversely by product number.
- Selects the first row that is returned, which is the one with the highest precedence, and returns its price.

This behavior is not suitable for our store since we want to receive realtime prices from our back-end system. We must replace the default behavior to meet our needs. We will write our own overridable function to do that.

19.4.2 Defining and Designing the New Behavior

Since our pricing algorithm is located on the back-end system, we want to use the exact algorithm on the Net.Commerce site. The back-end system uses its own tables to calculate product’s price. The correct approach, in our case, is to get the price from the back-end system by using the back-end pricing mechanism.

The Net.Commerce solution should look for reuse where possible. This reduces both the time and the cost to make the site. Many times, the back-end system does not have an existing API that Net.Commerce can use directly for integrating. For example, the price calculation can be done in an RPG procedure in an online transaction program. In this case, you have to invest more effort in integrating the back-end system and Net.Commerce.

In our example, we discovered that the existing back-end system has an API called GETPRICER, which we can use for realtime price retrieval. The default *GetBaseSpePrc_1.0(IBM,NC)* overridable function will be replaced by our new overridable function, which is called GETPRICE. GETPRICE will call the back-end API GETPRICER to return realtime pricing to the Net.Commerce *OrderDisplay* command. Figure 402 describes the elements of our solution.

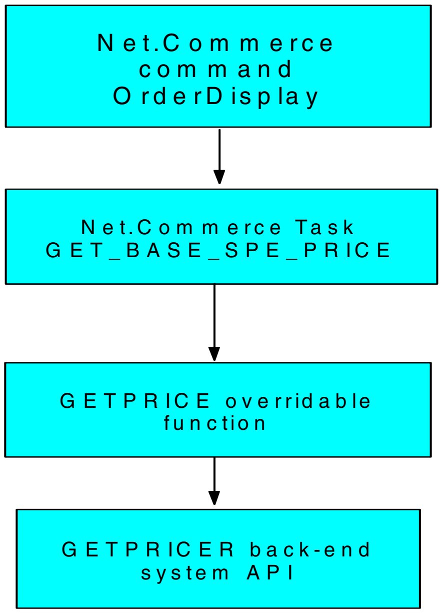

Net.Commerce command OrderDisplay
Net.Commerce Task GET_BASE_SPE_PRICE
GETPRICE overridable function
GETPRICER back-end system API

Figure 402. Price Retrieval Mechanism

This approach is only an example of Net.Commerce customizing. Net.Commerce is a complex product with many possible areas of customizing. The important point to remember is that if there is a change that you want to make, there is almost always a way to implement it.

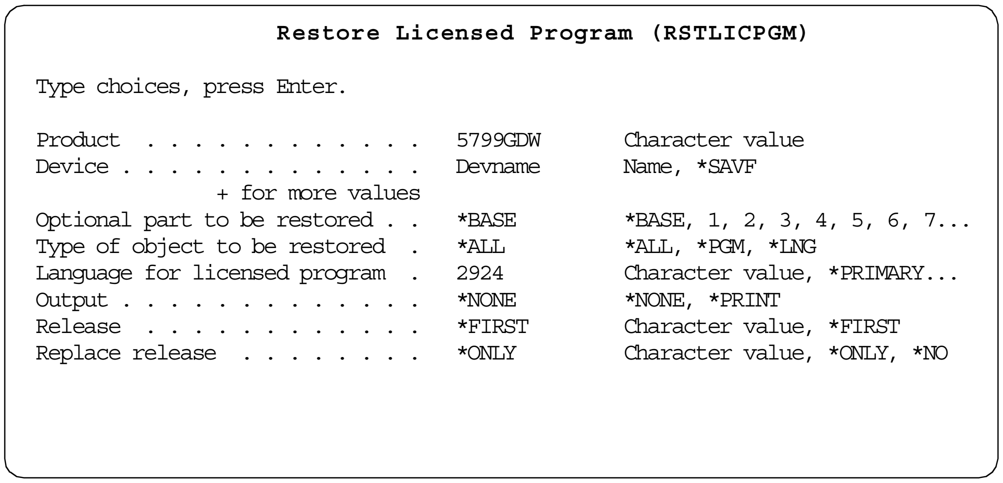
19.4.3 Coding the Overridable Function

We are now ready to set up the programming environment and code our overridable functions. This process is described in the following sections.

19.4.3.1 Setting Up the Environment

Overridable functions are an extension to the Net.Commerce server. They must be written in C++ because the server is written in C++ and has a specific interface to which the overridable functions should conform. As mentioned in 19.3.2, “Programming with C++” on page 417, use the C++ code only as a wrapper to your favorite language.

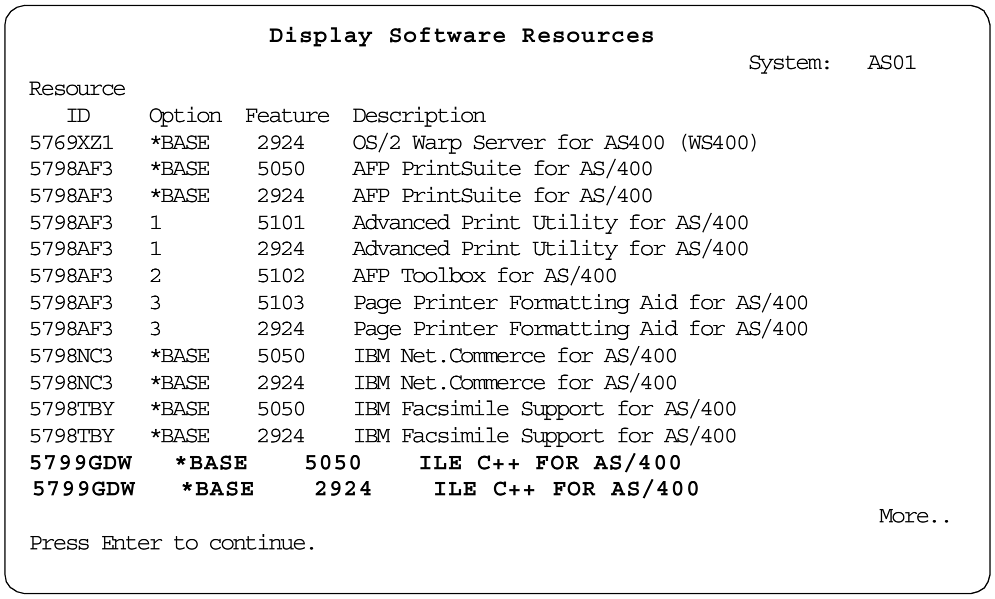
In our example, we used the C++ compiler PRPQ 5799GDW. Install 5799GDW using the `RSTLICPGM` command as shown in Figure 403 on page 421.



```
                    Restore Licensed Program (RSTLICPGM)

 Type choices, press Enter.

 Product  . . . . . . . . . . . .   5799GDW       Character value
 Device . . . . . . . . . . . . .   Devname       Name, *SAVF
                + for more values
 Optional part to be restored . .   *BASE         *BASE, 1, 2, 3, 4, 5, 6, 7...
 Type of object to be restored  .   *ALL          *ALL, *PGM, *LNG
 Language for licensed program  .   2924          Character value, *PRIMARY...
 Output . . . . . . . . . . . . .   *NONE         *NONE, *PRINT
 Release  . . . . . . . . . . . .   *FIRST        Character value, *FIRST
 Replace release  . . . . . . . .   *ONLY         Character value, *ONLY, *NO
```

Figure 403. Install the C++ PRPQ

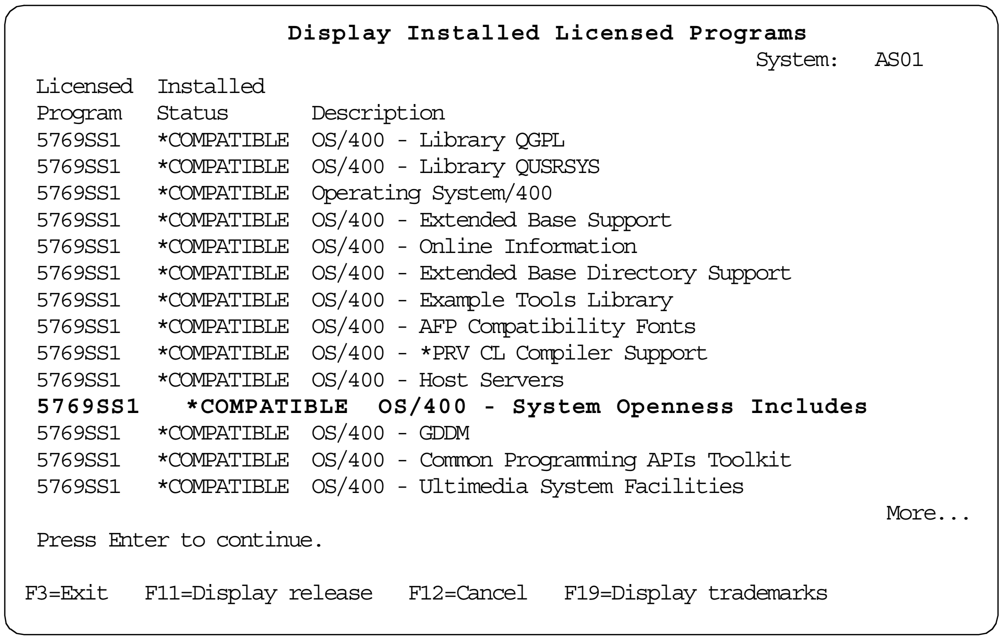
To verify your installation issue the command DSPSFWRSC. The command output should include the C++ PRPQ, as shown in Figure 404.



```
                    Display Software Resources
                                                          System:   AS01
 Resource
    ID      Option  Feature  Description
 5769XZ1    *BASE    2924    OS/2 Warp Server for AS400 (WS400)
 5798AF3    *BASE    5050    AFP PrintSuite for AS/400
 5798AF3    *BASE    2924    AFP PrintSuite for AS/400
 5798AF3    1        5101    Advanced Print Utility for AS/400
 5798AF3    1        2924    Advanced Print Utility for AS/400
 5798AF3    2        5102    AFP Toolbox for AS/400
 5798AF3    3        5103    Page Printer Formatting Aid for AS/400
 5798AF3    3        2924    Page Printer Formatting Aid for AS/400
 5798NC3    *BASE    5050    IBM Net.Commerce for AS/400
 5798NC3    *BASE    2924    IBM Net.Commerce for AS/400
 5798TBY    *BASE    5050    IBM Facsimile Support for AS/400
 5798TBY    *BASE    2924    IBM Facsimile Support for AS/400
 5799GDW    *BASE    5050    ILE C++ FOR AS/400
 5799GDW    *BASE    2924    ILE C++ FOR AS/400
                                                                  More..
 Press Enter to continue.
```

Figure 404. Verifying the C++ Installation

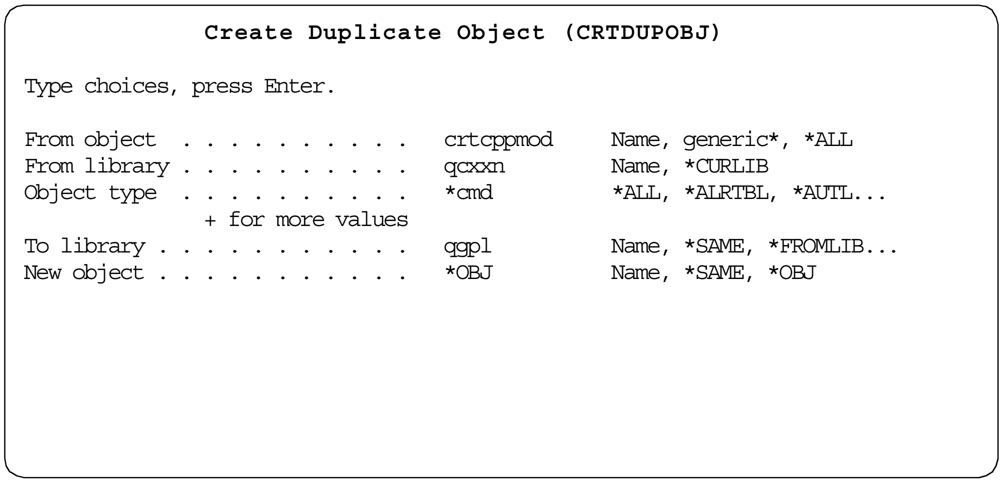
To compile your code, you need system openness includes, which are an optional part of the operating system. To verify that you have the includes, issue the command GO LICPGM and choose option **10** *(*Display installed licensed programs).



```
                    Display Installed Licensed Programs
                                                          System:   AS01
 Licensed  Installed
 Program   Status       Description
 5769SS1   *COMPATIBLE  OS/400 - Library QGPL
 5769SS1   *COMPATIBLE  OS/400 - Library QUSRSYS
 5769SS1   *COMPATIBLE  Operating System/400
 5769SS1   *COMPATIBLE  OS/400 - Extended Base Support
 5769SS1   *COMPATIBLE  OS/400 - Online Information
 5769SS1   *COMPATIBLE  OS/400 - Extended Base Directory Support
 5769SS1   *COMPATIBLE  OS/400 - Example Tools Library
 5769SS1   *COMPATIBLE  OS/400 - AFP Compatibility Fonts
 5769SS1   *COMPATIBLE  OS/400 - *PRV CL Compiler Support
 5769SS1   *COMPATIBLE  OS/400 - Host Servers
 5769SS1   *COMPATIBLE  OS/400 - System Openness Includes
 5769SS1   *COMPATIBLE  OS/400 - GDDM
 5769SS1   *COMPATIBLE  OS/400 - Common Programming APIs Toolkit
 5769SS1   *COMPATIBLE  OS/400 - Ultimedia System Facilities
                                                                    More...
 Press Enter to continue.

F3=Exit   F11=Display release   F12=Cancel   F19=Display trademarks
```

Figure 405. Verify the Openness Include

The C++ PRPQ is installed into library QCXXN. There are two compile commands in this library. The Create C++ Module (CRTCPPMOD) command creates C++ modules. The Create Binding C++ (CRTBNDCPP) command creates C++ programs. It is possible to duplicate the two commands to the library in your usual library list so that you will not have to add QCXXN to your library list. Duplicate the commands as shown in Figure 406.



```
               Create Duplicate Object (CRTDUPOBJ)

Type choices, press Enter.

From object  . . . . . . . . . .   crtcppmod     Name, generic*, *ALL
From library . . . . . . . . . .   qcxxn         Name, *CURLIB
Object type  . . . . . . . . . .   *cmd          *ALL, *ALRTBL, *AUTL...
               + for more values
To library . . . . . . . . . . .   qgpl          Name, *SAME, *FROMLIB...
New object . . . . . . . . . . .   *OBJ          Name, *SAME, *OBJ
```

Figure 406. Duplicating the Compile Commands

The C++ PRPQ primarily supports the AS/400 "root" file system. However, you can place your source code in the /QSYS.LIB file system. The files that are needed to create a command or overridable function should always be kept in one directory. We recommend that you place all directories for your overridable functions in the same parent directory. Figure 407 on page 423 displays the directory structure that we used in our example. It is found on the AS/400 IFS.

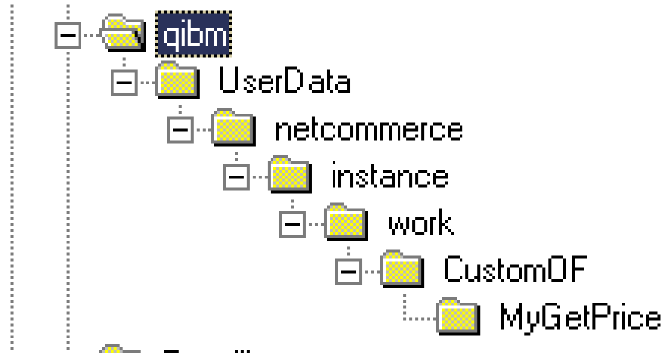


Figure 407. Directory Tree for Overridable Functions

To code the program, we mapped the integrated file system directory as a PC network drive and used a PC-based editor to edit files in that path. Another alternative is the use of the EDTF editor described in 7.3, “Stream File Handling Tools” on page 111.

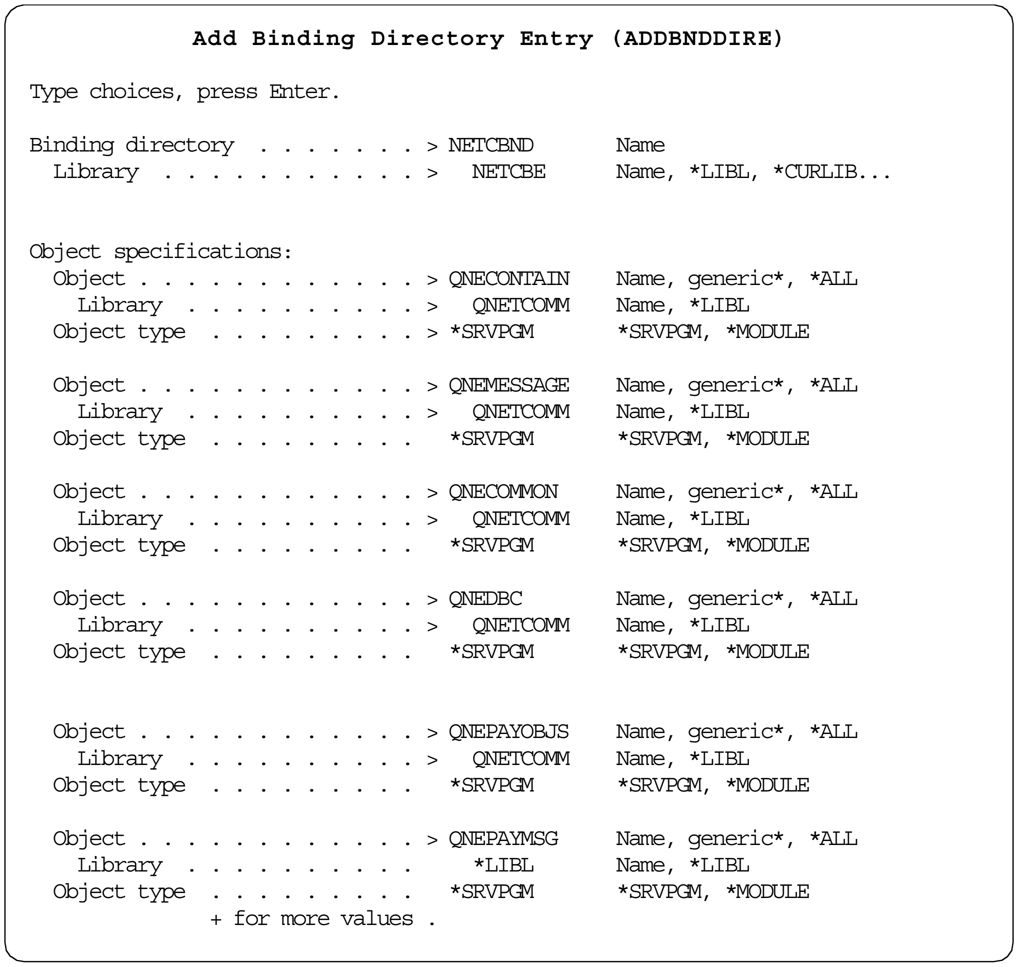
To properly link your overridable function, we recommend that you create a binding directory, which contains the Net.Commerce environment as described in Figure 408.

The first step is to create the binding directory. Type `CRTBNDDIR` and press **F4**.

```
                  Create Binding Directory (CRTBNDDIR)

 Type choices, press Enter.

 Binding directory  . . . . . . . > NETCBND       Name
   Library  . . . . . . . . . . . >   NETCBE      Name, *CURLIB
 Authority  . . . . . . . . . . .   *LIBCRTAUT    Name, *LIBCRTAUT, *CHANGE...
 Text 'description' . . . . . . . > 'Net.Commerce binding directory'
```

Figure 408. Creating Binding Directory for Net.Commerce

After the binding directory was created, add the relevant Net.Commerce service programs. Type `ADDBNDDIRE` and press **F4**. Type the service program names as shown on Figure 409 on page 424.



```
            Add Binding Directory Entry (ADDBNDDIRE)

Type choices, press Enter.

Binding directory  . . . . . . . > NETCBND       Name
  Library  . . . . . . . . . . . >   NETCBE      Name, *LIBL, *CURLIB...


Object specifications:
  Object . . . . . . . . . . . . > QNECONTAIN    Name, generic*, *ALL
    Library  . . . . . . . . . . >   QNETCOMM    Name, *LIBL
  Object type  . . . . . . . . . > *SRVPGM       *SRVPGM, *MODULE

  Object . . . . . . . . . . . . > QNEMESSAGE    Name, generic*, *ALL
    Library  . . . . . . . . . . >   QNETCOMM    Name, *LIBL
  Object type  . . . . . . . . .   *SRVPGM       *SRVPGM, *MODULE

  Object . . . . . . . . . . . . > QNECOMMON     Name, generic*, *ALL
    Library  . . . . . . . . . . >   QNETCOMM    Name, *LIBL
  Object type  . . . . . . . . .   *SRVPGM       *SRVPGM, *MODULE

  Object . . . . . . . . . . . . > QNEDBC        Name, generic*, *ALL
    Library  . . . . . . . . . . >   QNETCOMM    Name, *LIBL
  Object type  . . . . . . . . .   *SRVPGM       *SRVPGM, *MODULE


  Object . . . . . . . . . . . . > QNEPAYOBJS    Name, generic*, *ALL
    Library  . . . . . . . . . . >   QNETCOMM    Name, *LIBL
  Object type  . . . . . . . . .   *SRVPGM       *SRVPGM, *MODULE

  Object . . . . . . . . . . . . > QNEPAYMSG     Name, generic*, *ALL
    Library  . . . . . . . . . .     *LIBL       Name, *LIBL
  Object type  . . . . . . . . .   *SRVPGM       *SRVPGM, *MODULE
               + for more values .
```

Figure 409. Add the Net.Commerce Service Programs

19.4.3.2 Copying the Overridable Functions Skeleton Code

The C++ source shown in Figure 410 on page 425 is a skeleton for the Net.Commerce overridable function. An overridable function is like a program with one "sub-routine," which is called the *process routine*. The directory /qibm/proddata/netcommerce/adt/samples/tutorial contains some overridable function examples. The source code in Figure 410 on page 425 describes an empty overridable function.

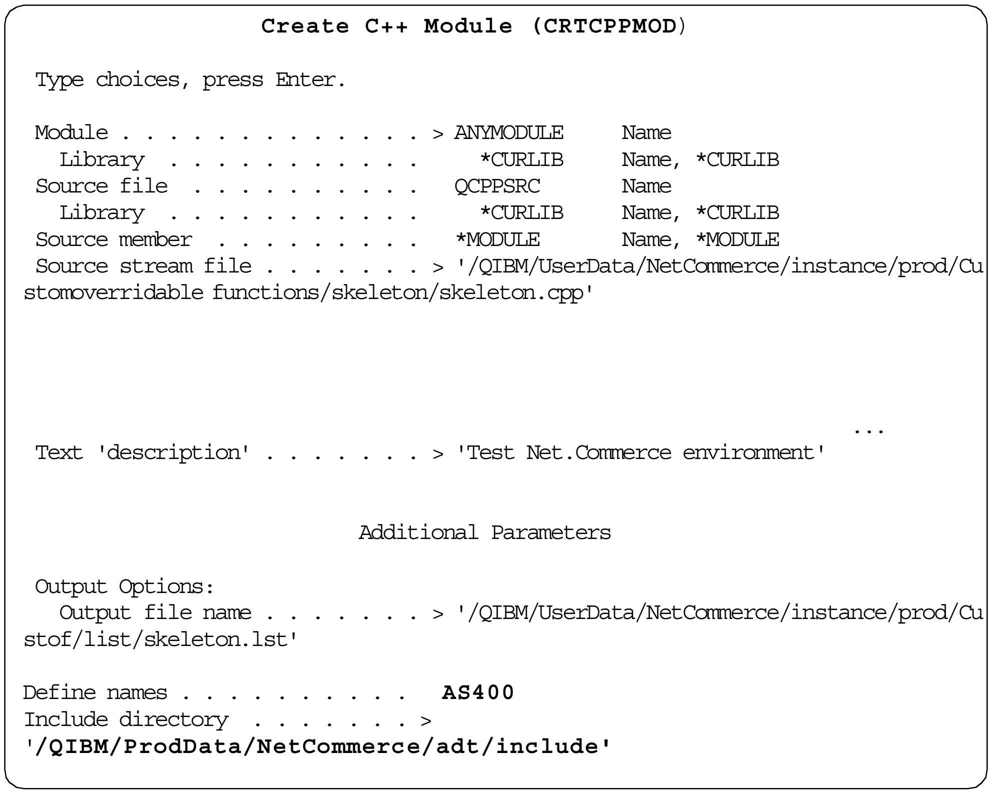
```
#ifdef AS400
#include "coibm.h"
#endif


#include "objects/objects.pch" (1)

#if defined(WIN32)
  #define __DLL_EXPORT__ __declspec(dllexport)
#else
  #define __DLL_EXPORT__
#endif


//#define __TRACE_NEW_OF__
#ifdef __TRACE_NEW_OF__
 typedef TraceYes Trace;
#else
 typedef TraceNo Trace;
#endif
static Trace trace ("NEW_OF ("__FILE__")");

class __DLL_EXPORT__ NewOF : public NC_OverridableFunction (2)
{
  public:
    NewOF() { }
    virtual  ~NewOF() { }(3)
    void operator delete (void *p){ ::delete p; }
    virtual void FailedRegistration (NC_RegistrationID &RegID, const ErrorMsg_Reg
*Err)
     { }
    virtual bool Process (const HttpRequest &Req, HttpResponse &Res, NC_Environment
&Env); (4)
      {
        return true;
      }
};
const ClassName NewOF::_STR_ThisClass("NewOF");
static bool X1 = NC_ApiManager::GetUniqueInstance().RegisterApi("IBM", "NC",
"NewOF", 1.0, new NewOF); (5)
```

Figure 410. Net.Commerc Overridable Functions Skeleton Code

Here are some comments about the skeleton (see the corresponding numbers in Figure 410):

1. The objects.pch include file has all the definitions for Net.Commerce.
2. This line defines the overridable functions. The overridable function name is NewOF.
3. The constructor and destructor is usually empty.
4. The process method is called by Net.Commerce. The function should return "true" for success and "false" for failure so that the correct error handler is invoked. The function communicates with the calling command by using the NC_Environment object.
5. This line of code is used by the Net.Commerce server to locate and initialize the overridable functions and must correspond to an entry in the overridable functions database table.

Now, try to compile the skeleton to verify the environment. Type `CRTCPPMOD` and press **F4**. The CRTCPPMOD command prompt is shown. Type the command parameters as shown in Figure 411 on page 426.



```
                    Create C++ Module (CRTCPPMOD)

 Type choices, press Enter.

 Module . . . . . . . . . . . . . > ANYMODULE     Name
   Library  . . . . . . . . . . .     *CURLIB     Name, *CURLIB
 Source file  . . . . . . . . . .   QCPPSRC       Name
   Library  . . . . . . . . . . .     *CURLIB     Name, *CURLIB
 Source member  . . . . . . . . .   *MODULE       Name, *MODULE
 Source stream file . . . . . . . > '/QIBM/UserData/NetCommerce/instance/prod/Cu
stomoverridable functions/skeleton/skeleton.cpp'





                                                                         ...
 Text 'description' . . . . . . . > 'Test Net.Commerce environment'


                           Additional Parameters

 Output Options:
   Output file name . . . . . . . > '/QIBM/UserData/NetCommerce/instance/prod/Cu
stof/list/skeleton.lst'

Define names . . . . . . . . . .  AS400
Include directory  . . . . . . . >
'/QIBM/ProdData/NetCommerce/adt/include'
```

Figure 411. Creating C++ Module Using the C++ PRPQ

The previous example creates the skeleton module, and names it ANYMODULE. The module is placed in the current library. The source to compile the module is read from an IFS file named /QIBM/UserData/NetCommerce/instance/prod/Customoverridable functions/skeleton/skeleton.cpp. The output of the compilation is written to an IFS file named skeleton.lst.

Table 23 briefly describes the main CRTCPPMOD command parameters.

Table 23. Command CRTCPPMOD Parameters

Parameter	**Parameter Description**
Modul (MODULE)	The name and library of the created module. In the Net.Commerce environment, this module will be linked to a service program.
Source file	Specifies the source physical file name and library of the file containing the C++ source code that you want to compile. All library, file, and member entries are converted to an Integrated File System filename before being processed.

Parameter	Parameter Description
Source stream file (SRCSTMF)	Specifies the path name of the stream file containing the C++ source code that you want to compile. The path name can be either absolutely or relatively qualified. An absolute path name starts with '/'; a relative path name starts with a character other than '/'. If absolutely qualified, the path name is complete. If relatively qualified, the path name is completed by pre-pending the job's current working directory to the path name. The SRCMBR or SRCFILE parameters cannot be specified with the SRCSTMF parameter.
Output file name(OUTPUT)	*NONE Does not generate the compiler listing. When a listing is not required, this default should be used because compile-time performance may improve. `file-name` — Specify the name of a basic IFS listing file. In our example, we used the file name skeleton.lst .
Include directory(INCDIR)	Specifies the name of one or more directories with include files. In the Net.Commerce environment, you must include the following directory on your include /QIBM/ProdData/NetCommerce/adt/include. In addition to the specified directory, the source directory is always searched for user include files.
Define names (DEFINE)	Specifies preprocessor macros that take affect before the file is processed by the compiler. When compiling in the Net.Commerce environment, the defined AS/400 system must be included in your define list.

19.4.4 Coding Your Overridable Function

In our example, the overridable function GETPRICE is simple. It should perform these actions:

- Get the product number from the environment
- Cast the environment variable and call the RPG program GETPRICER to retrieve realtime pricing from the back-end system
- Get the result from the RPG program and return it to the calling command by using the environment variable

Refer to Chapter 17, “Interfacing to Our Back-End Business System” on page 383, for a description of the back-end system tables involved in the price retrieval algorithm.

Figure 412 on page 428 and Figure 413 on page 429 show the GETPRICE overridable function code.

```
//*********************************************************/
// This function will call the back end system api in order */
// to calculate an item price.                           */
//                                                     */
// Author: shahar mor                                      */
// Provided AS IS                                          */
//*********************************************************/

#ifdef AS400           (1)
       #include "coibm.h"
#endif

#include <bcd.h>    // For working with packed decimal fields        (2)
#include "objects/objects.pch" // Net.Commerce include

// The following define is for compatibility with other platform compilers

#if defined(WIN32)
  #define __DLL_EXPORT__ __declspec(dllexport)
#else
  #define __DLL_EXPORT__
#endif


// Handling the trace option. Un remark the next line to enable trace.
//#define __TRACE_GetPrice__
#ifdef __TRACE_GET_TRACE__
 typedef TraceYes Trace;
#else
 typedef TraceNo Trace;
#endif

// Prototype the AS400 RPG program. We demonstrate OPM rpg program invocation.

extern "OS nowiden" void CalcPrice(char[],_DecimalT<7,0>,_DecimalT<7,0>,char[]);  (3)
#pragma map(CalcPrice ,"GETPRICER")

static Trace trace ("GetPrice ("__FILE__")");

class __DLL_EXPORT__ GetPrice : public NC_OverridableFunction
{
  public:
    GetPrice() { }

    virtual  ~GetPrice() { }

    void operator delete (void *p){ ::delete p; }

// Handle failed registration */          (4)

   virtual void FailedRegistration (NC_RegistrationID &RegID, const ErrorMsg_Reg *Err)
     {
       error << indent << "Error : GetPrice Registratio failed" << endl;
     }

// Main function */

   virtual bool Process (const HttpRequest &Req, HttpResponse &Res, NC_Environment &Env)
     {
        char ReturnPrice[14 ]={0}; // Return price from the RPG
        char InputSku[13] = "          "; // Input blank SKU to the RPG

// Defien the environment(We will only use the product number and return price  (5)
          static const StringWithOwnership
_PARAM_NAME_MERCHANT_REF_NUM("MERCHANT_REF_NUM");
     static const StringWithOwnership _PARAM_NAME_PRODUCT_REF_NUM("PRODUCT_REF_NUM");
      static const StringWithOwnership _PARAM_NAME_PRODUCT_PRICE("PRODUCT_PRICE");
          static const StringWithOwnership _PARAM_NAME_CURRENCY("CURRENCY");
```

Figure 412. GETPRICE Overridable Function (Part 1 of 2)

```
// Get function parameters from the env.       (6)
// 1. The merchant number
      String* MerchantRefNum = (String*) Env.Seek(_PARAM_NAME_MERCHANT_REF_NUM);
        if (MerchantRefNum == NULL)
        {
             error << indent << "Error : Cant get merchant number    " << endl;
               return false;
        }
// 2. The product number. This is Net.Commerce number. we will derive the real SKU
//    number in the RPG program.

      String* ProductRefNum = (String*) Env.Seek(_PARAM_NAME_PRODUCT_REF_NUM);
        if (ProductRefNum == NULL)
        {
             error << indent << "Error : Cant get product number     " << endl;
               return false;
        }

// 3.  Get the price. This will contain the returned price from the RPG program

      String* ProductPrice  = (String*) Env.Seek(_PARAM_NAME_PRODUCT_PRICE);
        if (ProductPrice  == NULL)
        {
             error << indent << "Error : Cant get product price point " << endl;
               return false;
        }

// 4.  Get the currency. You can return currency information here(Not used in our example)

       String* Currency   = (String*) Env.Seek(_PARAM_NAME_CURRENCY);
        if (Currency  == NULL)
        {
             error << indent << "Error : Cant get Currency info      " << endl;
               return false;
        }

// The following 2 lines handle packed decimal. it will convert string to packed  (7 )

      _DecimalT<7,0> ProductNumber = __D((char *) ProductRefNum->c_str());
       _DecimalT<7,0> CustomerNumber = __D("0");

// Call the RPG program to get real time pricing
      CalcPrice(InputSku,ProductNumber,CustomerNumber,ReturnPrice);

       *ProductPrice << ReturnPrice;                 (8)
       return true;
}
};
static bool X2 = NC_ApiManager::GetUniqueInstance().RegisterApi("IBM", "NC",
                             "GetPrice", 1.0, new GetPrice);
```

Figure 413. GETPRICE Overridable Function (Part 2 of 2)

Note these remarks for the GetPrice source code:

1. The coibm.h is included because we will compile the code with the #define AS/400.
2. The bcd.h include is required in our program due to the fact that the RPG program GETPRICER works with packed decimal parameters.
3. The RPG program GETPRICER is prototyped. The prototype in our example is suitable for all OPM programs.
4. The failed registration routine is optional. Net.Commerce calls this function in case of failure to register the overridable function. In our example, we simply documented the failure.

5. The overridable function interacts with the calling program through environment variables. Our example uses only part of the available environment variables.
6. The env.seek operation actually receives the value of the requested environment variable.
7. The __D constructor converts strings to a packed decimal. Our code assumes the validity of the input parameters since the calling command performed the check parameters function.
8. ReturnPrice is calculated in the RPG program, and we set the price environment variable to the calculated price.

Important

Keep these points in mind:

- Do not return number strings with a comma. For example, the string 1,235.24 will not be treated as a number by Net.Commerce.
- Our RPG program appended null to the result price. Failure to append null to the result string causes problems.
- If an update to the Net.Commerce tables is required, do not use native data base support to perform it. Use the Net.Commerce database class methods to update Net.Commerce tables.

The back-end system calculates the price using the RPG program GETPRICER. The GETPRICER program receives the product number from the overridable function, converts the Net.Commerce product number to the back-end system product number, and looks up the product price in the back-end product and discount table. Then, it calculates the price for the "Web customer" and returns the price to the GETPRICE overridable function.

To make our example complete, we list the GETPRICER source code in Figure 414 on page 431 through Figure 416 on page 433.

```
H*******************************************************************************
   H* This program demonstrates the API in the existing system for price calculation
    H*
    H* Parms:
    H*      InSku   - Product SKU in the back end system. If coming
    H*              from Net.Commerce this parameter will be blank.
    H*      InProd  - Product number. This is the product number in
    H*               the net commerce system.
    H*              if this program is called from regular back end
    H*               operation this parameter will be 0.
    H*               product number.
    H*      InCust  - The customer number to calculate price for.
    H*
    H*      OuPric  - The calculated price. The price will be null
    H*              terminated if the request was originated by
    H*               Net.Commerce
    H*
    H* Files:
    H*      BEPROD  - Back end system products file
    H*      BEDISC  - Back end system discounts file
    H*      PRODUCT - Net.Commerce product file
    H*
    H*
    H* Author: Shahar mor
    H* Provided AS IS
 H*******************************************************************************
    H*
     HDFTACTGRP(*NO)    CTGRP(*CALLER)  ALWNULL(*INPUTONLY)            (1)
    F*
    F*  Products File
    FBEPROD   IF   E          K DISK    RENAME(BEPROD:RBEPROD)
    F
    F*  Discounts file
    FBEDISC   IF   E          K DISK    RENAME(BEDISC:RBEDISC)
    F
    F*  Net.Commerce product file
    FPRODUCT  IF   E          K DISK    RENAME(PRODUCT:RPRODUCT)    (2)
    D*
    FLOGNETC   O    E           DISK                              (3)
    D*  Parameter Structure
    D  InSku        S           12                               (4)   Product number
    D  InProd       S            7  0                                  Product number
    D  InCust       S            7  0                                  Customer Number
    D  OuPric       S           14                                     Returned Price
    D*
    D*  Keys for discount file
    D KySku         S           12
    D KyCust        S            7  0
    D*
    D*  Key for Net.Commerce product file
    D               DS
    D KyNprod              1     4B 0             (5)
    D*
    D*  Stand alone working fields
    D  CPric        S           13  2                                  Returned Price
    D  Error        S            1                                     Returned Price
    D  CurProd      S            7  0
    D  Len          S            3  0
    D*
    D*  Constants
    D  Null         C                CONST(X'00')                       'X'00'
    D  Regular      C                 CONST(0 )
    D*
```

Figure 414. GETPRICER RPG Program Source Code (Part 1 of 3)

```
C*******************************************************************************
  C*                  Main logic
 C*******************************************************************************
  C*
  C     *ENTRY        PLIST
  C                   PARM                    InSku
  C                   PARM                    InProd
  C                   PARM                    InCust
  C                   PARM                    OuPric
  C*
 C     KyDisc        KLIST
  C                   KFLD                    KySku
  C                   KFLD                    KyCust
  C
  C*  Request from Net.Commerce  ?
  C                   EVAL      OuPric = *BLANK
  C                   EVAL      Error = *OFF
  C                   IF        InProd <> Regular
  C                   EXSR      GetSku
  C                   ELSE
  C                   EVAL      KySku   = InSku
  C                   ENDIF
  C
  C*  Check for product existence
  C     KySku         CHAIN(E)  RBEPROD
  C                   IF        Not %found
  C                   MOVE      *ON           Error
  C                   ENDIF
  C*
  C                   IF        Error = *OFF
  C*
  C                   IF        InProd <> Regular
  C                   Eval      KyCust = 0
  C                   ELSE
  C                   Eval      KyCust = InCust
  C                   ENDIF
  C*
  C                   Eval      CPric = BePric
  C*
  C     KyDisc        CHAIN(E)  RBEDISC
  C                   IF        %Found
  C                  EVAL     CPric = CPric * (100 - BdPct) / 100
  C                   ENDIF
  C
  C                   IF        InProd <> Regular
  C                  EVAL     Oupric = %trim(%editc(Cpric:'3'))
  C     ' '           CHECKR    OuPric        Len
  C                  Eval     %subst(Oupric:Len + 1:1) = Null     (6)
  C                   ELSE
  C                  EVAL     Oupric = %trim(%editc(Cpric:'3'))
  C                   ENDIF
  C*
  C                   ELSE
  C*
  C                   IF        InProd <> Regular
  C                   EVAL      Oupric = Null
  C                   ELSE
  C                   EVAL      Oupric = *BLANK
  C                   ENDIF
  C*
  C                   ENDIF
  C*
  C                   EXSR      LOGRQS
  C                   RETURN
```

Figure 415. GETPRICER RPG Program Source Code (Part 2 of 3)

```
C*************************************************************
 C    GetSku       BEGSR
C*************************************************************
 C*
 C*  Get the back end product number from the Net.Commerce number
 C*
 C*
 C                 Z-ADD     InProd       KyNprod
 C    KyNProd       CHAIN     RPRODUCT
 C                 IF        %Found
 C                 EVAL      KySku = %subst(PRNBR:1:12)
 C                 ELSE
 C                 EVAL      KySKU = *BLANK
 C                 ENDIF
 C                 ENDSR
C*************************************************************
 C    LOGRQS       BEGSR
C*************************************************************
 C*
 C                 EVAL      XnProd = InProd
 C                 EVAL      XnCust = XnCust
 C                 EVAL      XnSku  = InSku
 C                 EVAL      XnPric = OuPric
 C                 WRITE     RLOG
 C                 ENDSR
```

Figure 416. GETPRICER RPG Program Source Code (Part 3 of 3)

Note these remarks in regard to the GETPRICER program (the numbers correspond to those shown in Figure 414 on page 431 through Figure 416):

1. The RPG program has to map the product number from the Net.Commerce product number to the back-end system product number. To perform the map operation, the RPG program has to read from the PRODUCTS table in Net.Commerce. The PRODUCT table contains null capable columns so we ask the RPG program to allow null by specifying `ALWNULL(*INPUTONLY)`.
2. The product table is the Net.Commerce main product table. We use this table to map the Net.Commerce product number to our back-end system product number.
3. The log table is optional. We used this table to track requests coming to the GETPRICER API.
4. The input parameters must match the parameter definition in the calling overridable function GETPRICE. For example, the customer number is packed with a length (7,0). It is important to see that you can easily pass parameters from and to the C++ GETPRICE.
5. To perform a chain operation to the Net.Commerce product table, we must define the key as binary. Most of the Net.Commerce keys will be of type integer, which requires bin(4) in the RPG D specifications.
6. The returned price is eventually passed by the overridable function to Net.Commerce using environment variables. Environment variables are always null terminated strings. We used the %EDITC built-in function to convert the numeric price to a character string and then appended null in the end of the string. We converted to string in the RPG program. The conversion can also be done on the calling C++ program.

19.4.5 Compiling the Overridable Function

You must now compile your overridable function and add it to a service program. Figure 417 shows the CL code segment to do this.

```
/***************************************************************/
/* Create C++ module and service program for Net.Commerce      */
/*                                                             */
/* Uses source from stmf                                       */
/***************************************************************/
             PGM        PARM(&Stmf &Function)

             DCL        VAR(&Stmf   )   TYPE(*CHAR) LEN(64)
             DCL        VAR(&Function) TYPE(*CHAR) LEN(10)
             DCL        VAR(&Text    ) TYPE(*CHAR) LEN(50)

             CRTCPPMOD  MODULE(QTEMP/&FUNCTION) SRCSTMF(&STMF) +
                           DEFINE(AS400) +
                          INCDIR('/Qibm/Proddata/netcommerce/adt/incl+
                          ude')
             CHGVAR     VAR(&TEXT) VALUE('Implementation of +
                          Net.Commerce OF-' *BCAT &function)
             CRTSRVPGM  SRVPGM(QNETCOMM/&FUNCTION) +
                          MODULE(QTEMP/&function) EXPORT(*ALL) +
                          TEXT(&text) BNDDIR(NETCBND)
             MONMSG     MSGID(CPF0000)

             ENDPGM
```

Figure 417. Compile the GETPRICE Service Program

The following compile created the module in the library QTEMP and with no debug information. You may want to include debug information when you are in the development stage of your overridable function as described in 19.5.1, "Compiling the Overridable Function with Debug Information" on page 437. The service program you created is loaded by the Net.Commerce startup process.

19.4.6 Registering the Overridable Function in the Database

For an overridable function to exist in the system, it has to be registered in the database. Otherwise, Net.Commerce will not know that the OF exists and will not load it on the server startup. Currently, this registration is done by manually inserting a row into the OFS table as shown here:

```
insert into ofs (refnum,dll_name,vendor,product,name,version, description)
Select max(refnum) + 1, 'serviceprogram'
,'IBM','NC','overridablefunctionname', 1.0, 'Some description' from ofs
```

In Appendix A, "Source Code Samples" on page 463, we provide an example of the REGOFS command for overridable function database registration. Refer to *Net.Commerce Commands, Tasks, Overridable Functions, and Database Tables* for a detailed description of the OFS table.

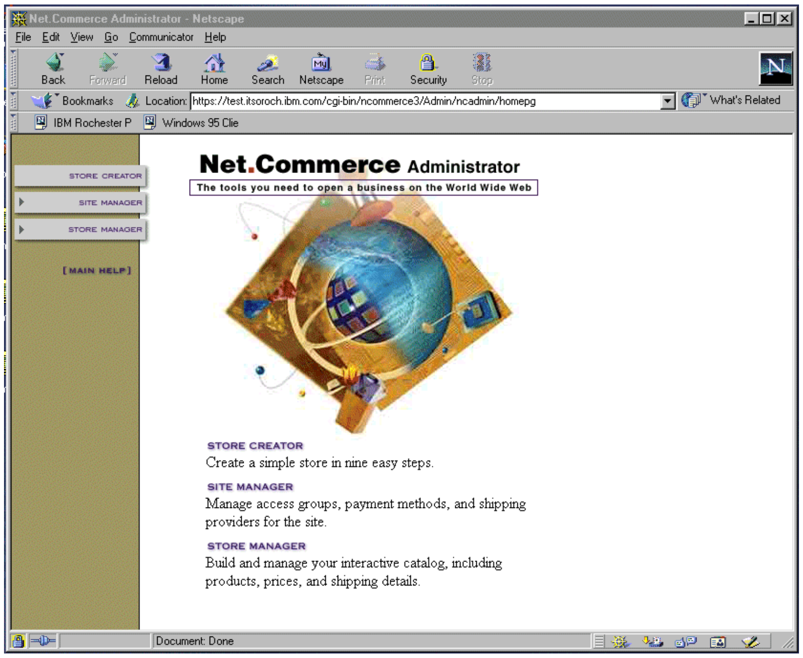
19.4.7 Assigning the Overridable Function

Once the overridable function is in the OFs table, you can assign the overridable function to a task by using ncadmin forms. To assign the overridable functions, complete the following steps:

1. Select the Net.Commerce administration URL from your browser, for example http://myserver/ncadmin/.

2. The Net.Commerce administrator is shown. Type the administrator name and password. Click on **Logon**.

   The Net.Commerce administrator window appears in your browser as shown in Figure 418.


Net.Commerce Administrator - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: https://test.itsoroch.ibm.com/cgi-bin/ncommerce3/Admin/ncadmin/homepg
What's Related
IBM Rochester P Windows 95 Clie
STORE CREATOR
SITE MANAGER
STORE MANAGER
[MAIN HELP]
Net.Commerce Administrator
The tools you need to open a business on the World Wide Web
Document: Done

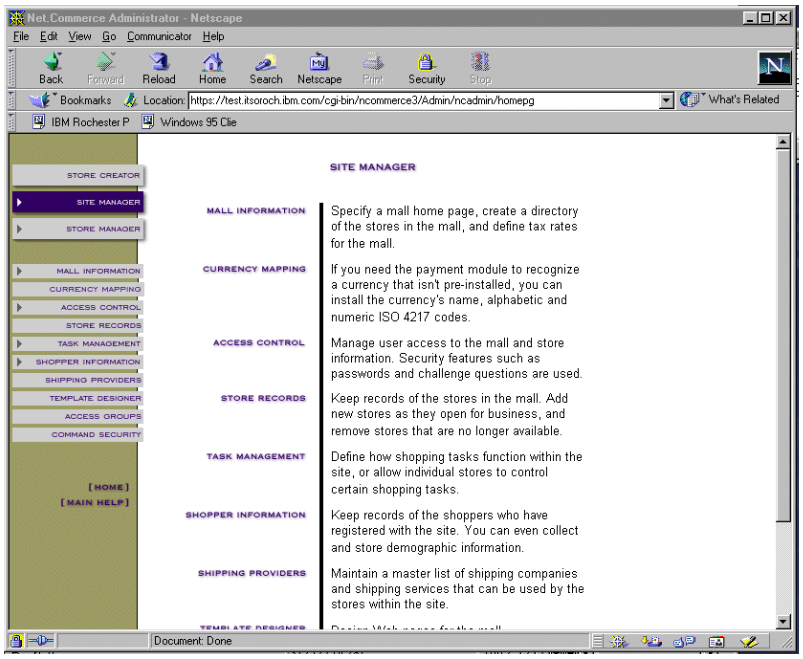
STORE CREATOR
Create a simple store in nine easy steps.

SITE MANAGER
Manage access groups, payment methods, and shipping providers for the site.

STORE MANAGER
Build and manage your interactive catalog, including products, prices, and shipping details.

Figure 418. Net.Commerce Administrator Window

3. Click on **Site Manager**. The Net.Commerce site manager window appears as shown in Figure 419.

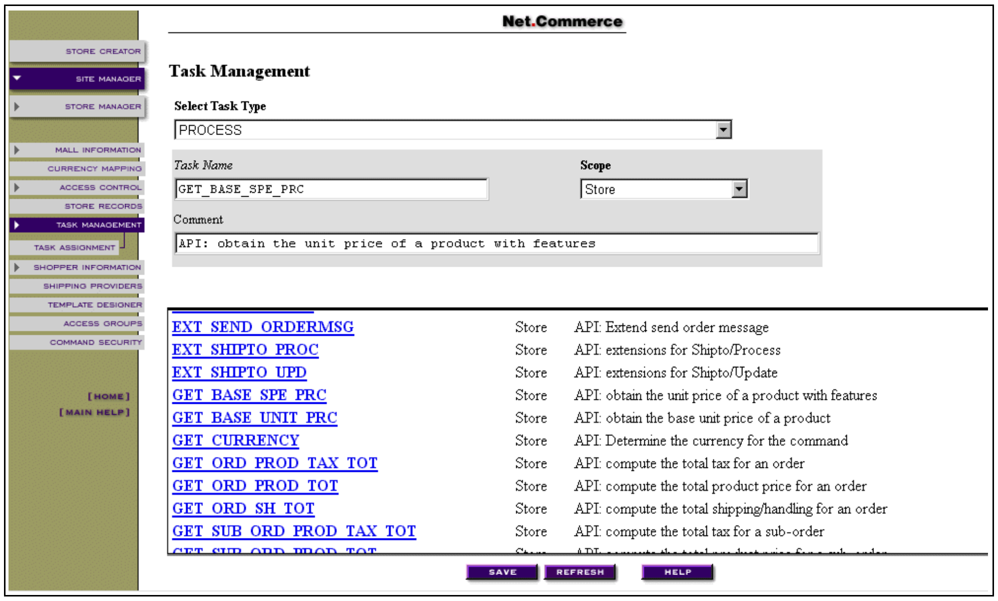

Net.Commerce Administrator - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: https://test.itsoroch.ibm.com/cgi-bin/ncommerce3/Admin/ncadmin/homepg
What's Related
IBM Rochester P Windows 95 Clie
STORE CREATOR
SITE MANAGER
STORE MANAGER
MALL INFORMATION
CURRENCY MAPPING
ACCESS CONTROL
STORE RECORDS
TASK MANAGEMENT
SHOPPER INFORMATION
SHIPPING PROVIDERS
TEMPLATE DESIGNER
ACCESS GROUPS
COMMAND SECURITY
[HOME]
[MAIN HELP]
Document: Done

SITE MANAGER

MALL INFORMATION	Specify a mall home page, create a directory of the stores in the mall, and define tax rates for the mall.
CURRENCY MAPPING	If you need the payment module to recognize a currency that isn't pre-installed, you can install the currency's name, alphabetic and numeric ISO 4217 codes.
ACCESS CONTROL	Manage user access to the mall and store information. Security features such as passwords and challenge questions are used.
STORE RECORDS	Keep records of the stores in the mall. Add new stores as they open for business, and remove stores that are no longer available.
TASK MANAGEMENT	Define how shopping tasks function within the site, or allow individual stores to control certain shopping tasks.
SHOPPER INFORMATION	Keep records of the shoppers who have registered with the site. You can even collect and store demographic information.
SHIPPING PROVIDERS	Maintain a master list of shipping companies and shipping services that can be used by the stores within the site.

Figure 419. Net.Commerce Site Manager Window

4. Click on **Task Management** in the left frame. The Net.Commerce Task Management window is displayed as shown in Figure 420.



Task Management

Select Task Type: PROCESS

Task Name: GET_BASE_SPE_PRC

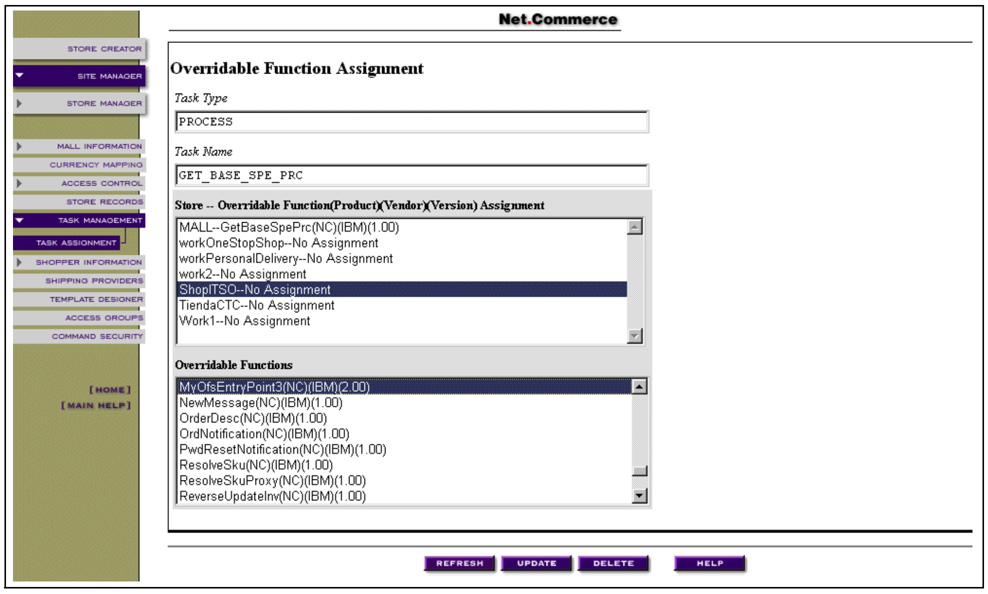
Scope: Store

Comment: API: obtain the unit price of a product with features

EXT_SEND_ORDERMSG	Store	API: Extend send order message
EXT_SHIPTO_PROC	Store	API: extensions for Shipto/Process
EXT_SHIPTO_UPD	Store	API: extensions for Shipto/Update
GET_BASE_SPE_PRC	Store	API: obtain the unit price of a product with features
GET_BASE_UNIT_PRC	Store	API: obtain the base unit price of a product
GET_CURRENCY	Store	API: Determine the currency for the command
GET_ORD_PROD_TAX_TOT	Store	API: compute the total tax for an order
GET_ORD_PROD_TOT	Store	API: compute the total product price for an order
GET_ORD_SH_TOT	Store	API: compute the total shipping/handling for an order
GET_SUB_ORD_PROD_TAX_TOT	Store	API: compute the total tax for a sub-order

Figure 420. Select Task for Assignment

5. Select the **Process** as the *Task Type*. Enter the *task name* as get_base_spe_prc. Select **Store** for the *scope* of the change we will make.
6. Click on the **Task Assignment** link on the left side of the display. The Overridable Function Assignment window appears (Figure 421).



Overridable Function Assignment

Task Type: PROCESS

Task Name: GET_BASE_SPE_PRC

Store -- Overridable Function(Product)(Vendor)(Version) Assignment

```
MALL--GetBaseSpePrc(NC)(IBM)(1.00)
workOneStopShop--No Assignment
workPersonalDelivery--No Assignment
work2--No Assignment
ShopITSO--No Assignment
TiendaCTC--No Assignment
Work1--No Assignment
```

Overridable Functions

```
MyOfsEntryPoint3(NC)(IBM)(2.00)
NewMessage(NC)(IBM)(1.00)
OrderDesc(NC)(IBM)(1.00)
OrdNotification(NC)(IBM)(1.00)
PwdResetNotification(NC)(IBM)(1.00)
ResolveSku(NC)(IBM)(1.00)
ResolveSkuProxy(NC)(IBM)(1.00)
ReverseUpdateInv(NC)(IBM)(1.00)
```

Figure 421. Assign Task to Overridable Function

7. Select the store to which to assign the overridable function. Then, from the bottom of the window, select the overridable function **GETPRICE** that we wrote. Click on the **UPDATE** button.

In our example, the store ShopITSO implements the task GET_BASE_SPE_PRC by using the overridable function GetPrice. Prices are retrieved from the back-end system.

19.5 Testing and Debugging the Overridable Function

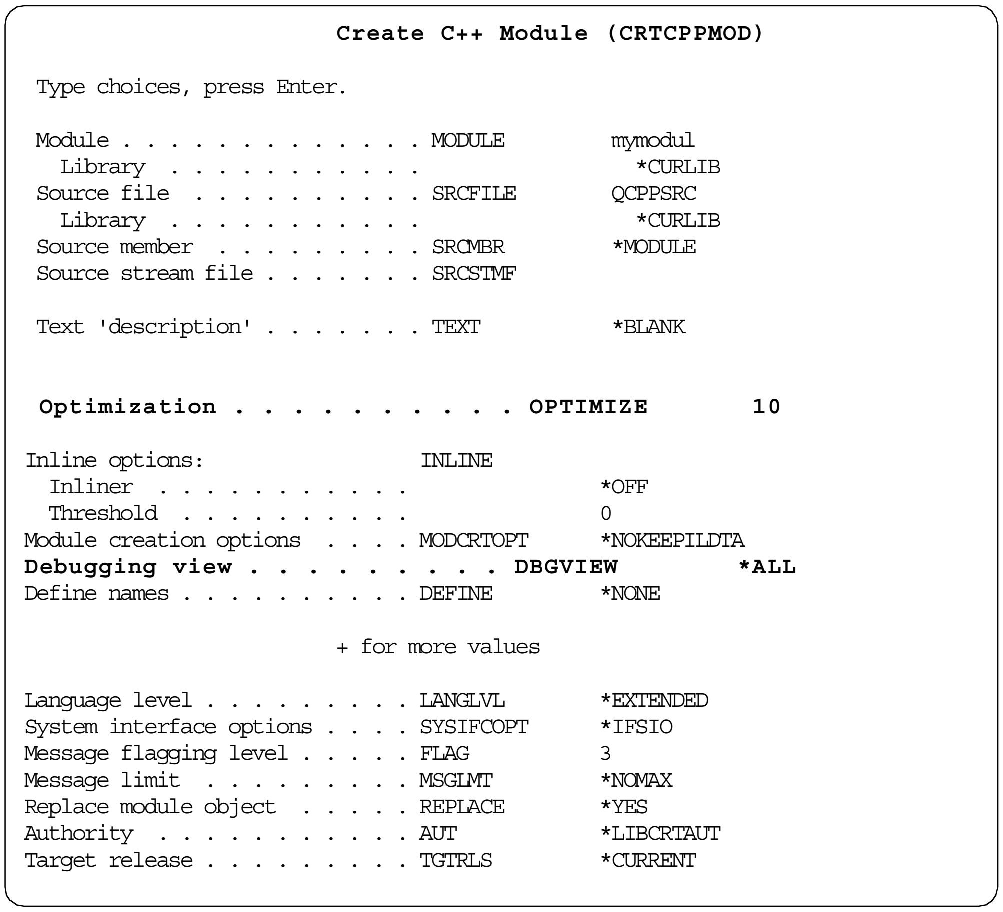
In your site implementation, you will most likely use a test instance for developing and testing new commands and overridable functions. Once your overridable function is compiled and registered, restart the Net.Commerce server to let Net.Commerce load it. After Net.Commerce is restarted, it is time to test and debug the overridable function. Debugging the overridable function can be tricky since the function can run on any of the Net.Commerce server jobs.

To test and debug the overridable function, complete these tasks:

1. Compile the function modules with the debug information.
2. Identify the command that activates the task, which invokes the overridable function. Start the service job for each Net.Commerce job.
3. Start debug.

19.5.1 Compiling the Overridable Function with Debug Information

The compiled source must include debug information to be debugged. The optimization level must be set to a minimum level for variables to be displayed and modified while debugging. Figure 422 on page 438 shows the parameters that you must specify on the CRTCPPMOD screen to enable debug on your overridable function.



```
                         Create C++ Module (CRTCPPMOD)

 Type choices, press Enter.

 Module . . . . . . . . . . . . . MODULE        mymodul
   Library  . . . . . . . . . . .                 *CURLIB
 Source file  . . . . . . . . . . SRCFILE       QCPPSRC
   Library  . . . . . . . . . . .                 *CURLIB
 Source member  . . . . . . . . . SRCMBR        *MODULE
 Source stream file . . . . . . . SRCSTMF

 Text 'description' . . . . . . . TEXT          *BLANK


 Optimization . . . . . . . . . . OPTIMIZE       10

Inline options:                  INLINE
  Inliner  . . . . . . . . . . .                 *OFF
  Threshold  . . . . . . . . . .                 0
Module creation options  . . . . MODCRTOPT       *NOKEEPILDTA
Debugging view . . . . . . . . . DBGVIEW         *ALL
Define names . . . . . . . . . . DEFINE          *NONE

                          + for more values

Language level . . . . . . . . . LANGLVL         *EXTENDED
System interface options . . . . SYSIFCOPT       *IFSIO
Message flagging level . . . . . FLAG            3
Message limit  . . . . . . . . . MSGLMT          *NOMAX
Replace module object  . . . . . REPLACE         *YES
Authority  . . . . . . . . . . . AUT             *LIBCRTAUT
Target release . . . . . . . . . TGTRLS          *CURRENT
```

Figure 422. Create C++ Module with Debug Information

Note

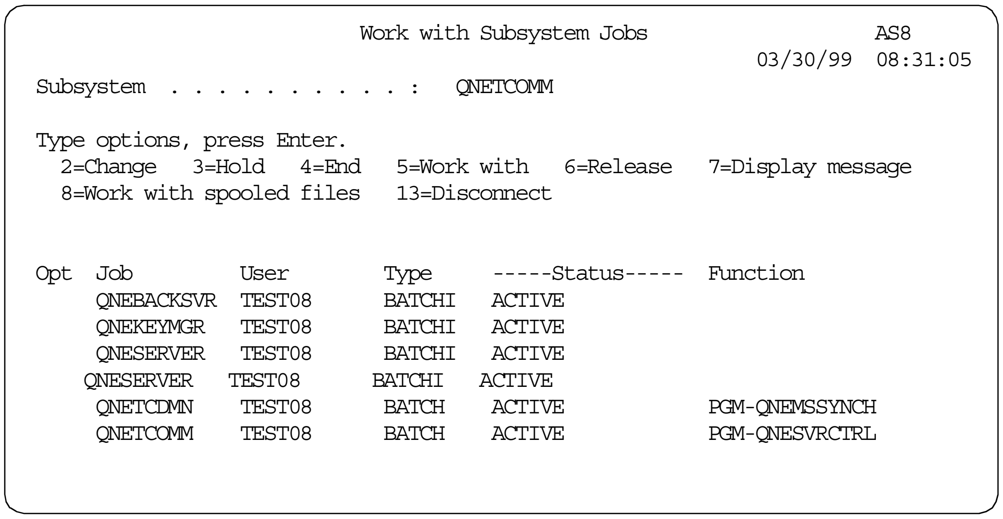
Do not forget to set the optimization level to "40" and the debug view to "none" after finishing your debug.

In our example, we called the RPG program GETPRICER to retrieve pricing. To enable source debugging on the RPG program, set the DBGVIEW parameter on the CRTBNDRPG command to *LIST or *SOURCE.

19.5.2 Starting Net.Commerce Service Jobs

Since the overridable function will run under Net.Commerce processes, issue the `STRSRVJOB` command to each of the Net.Commerce servers. Complete these steps:

1. Identify Net.Commerce jobs details. Issue the `WRKSBSJOB QNETCOMM` command to display the Net.Commerce job. Figure 423 on page 439 shows the WRKSBSJOB screen.

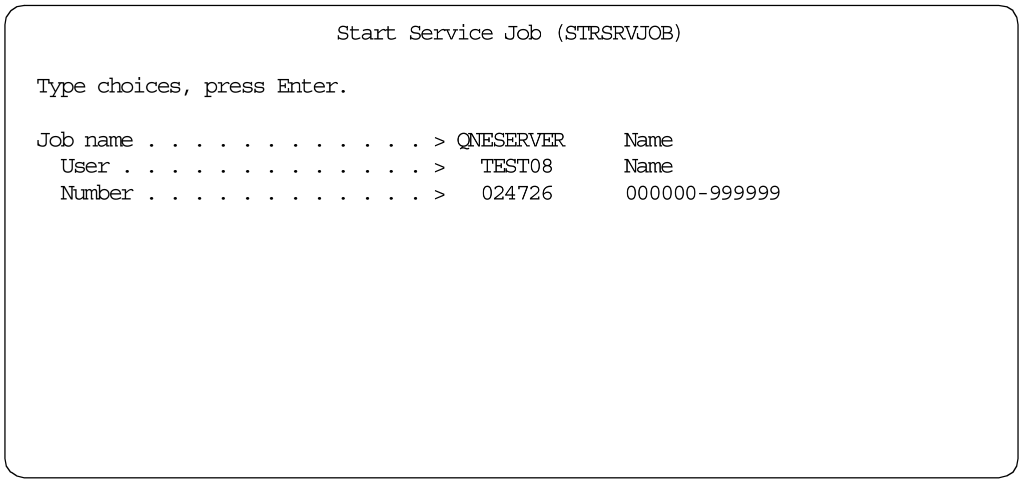


```
                          Work with Subsystem Jobs                   AS8
                                                       03/30/99  08:31:05
 Subsystem  . . . . . . . . . . :   QNETCOMM

 Type options, press Enter.
   2=Change   3=Hold   4=End   5=Work with   6=Release   7=Display message
   8=Work with spooled files   13=Disconnect


 Opt  Job         User        Type     -----Status-----  Function
      QNEBACKSVR  TEST08      BATCHI   ACTIVE
      QNEKEYMGR   TEST08      BATCHI   ACTIVE
      QNESERVER   TEST08      BATCHI   ACTIVE
     QNESERVER   TEST08      BATCHI   ACTIVE
      QNETCDMN    TEST08      BATCH    ACTIVE            PGM-QNEMSSYNCH
      QNETCOMM    TEST08      BATCH    ACTIVE            PGM-QNESVRCTRL
```

Figure 423. Display Net.Commerce Jobs

2. Write down the details for each of the QNESERVER jobs with the instance name that you wish to debug. For each job, issue the `STRSRVJOB` command. Each STRSRVJOB command must be issued from a different session so you must open more than one session on your PC.

   Figure 424 displays an example to the STRSRVJOB command prompt.



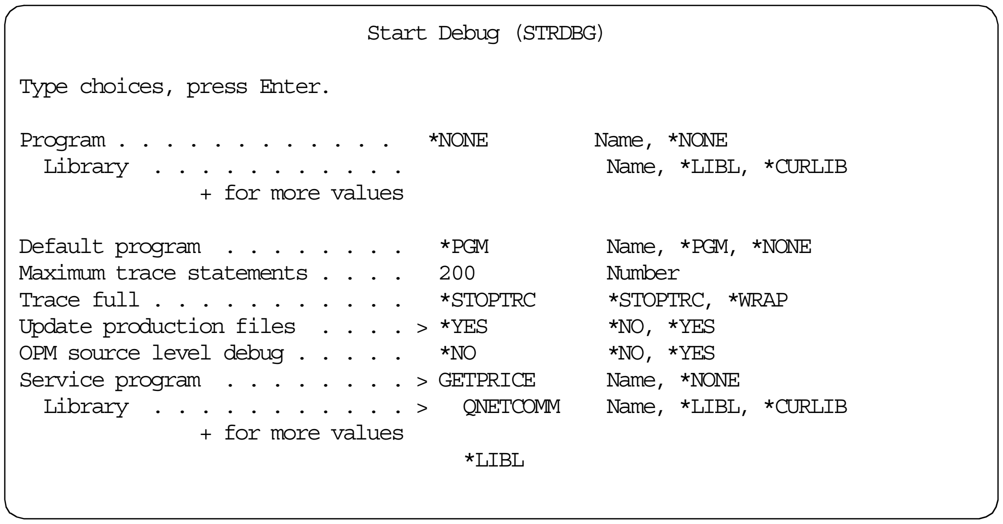
```
                          Start Service Job (STRSRVJOB)

 Type choices, press Enter.

 Job name . . . . . . . . . . . . . > QNESERVER     Name
   User . . . . . . . . . . . . . . >   TEST08      Name
   Number . . . . . . . . . . . . . >   024726      000000-999999
```

Figure 424. Start Servicing Net.Commerce Process

19.5.3 Start Debug

We can now start debug from the sessions that issued the STRSRVJOB command:

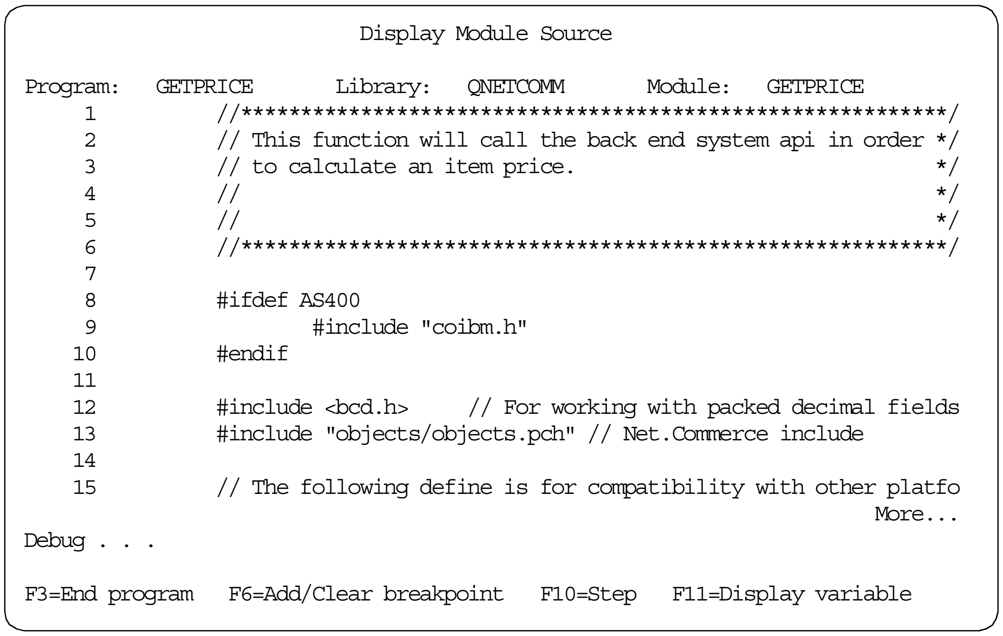
1. Type the `STRDBG` command.
2. Press the **F4** key and fill the parameters as described in Figure 425 on page 440.



```
                             Start Debug (STRDBG)

Type choices, press Enter.

Program . . . . . . . . . . . .   *NONE         Name, *NONE
  Library  . . . . . . . . . . .                 Name, *LIBL, *CURLIB
               + for more values

Default program  . . . . . . . .   *PGM          Name, *PGM, *NONE
Maximum trace statements . . . .   200           Number
Trace full . . . . . . . . . . .   *STOPTRC      *STOPTRC, *WRAP
Update production files  . . . . > *YES          *NO, *YES
OPM source level debug . . . . .   *NO           *NO, *YES
Service program  . . . . . . . . > GETPRICE      Name, *NONE
  Library  . . . . . . . . . . . >   QNETCOMM    Name, *LIBL, *CURLIB
               + for more values
                                     *LIBL
```

Figure 425. Start Debug the Overridable Function

3. The display module source screen appears as shown in Figure 426. Press **F10** to leave this screen.

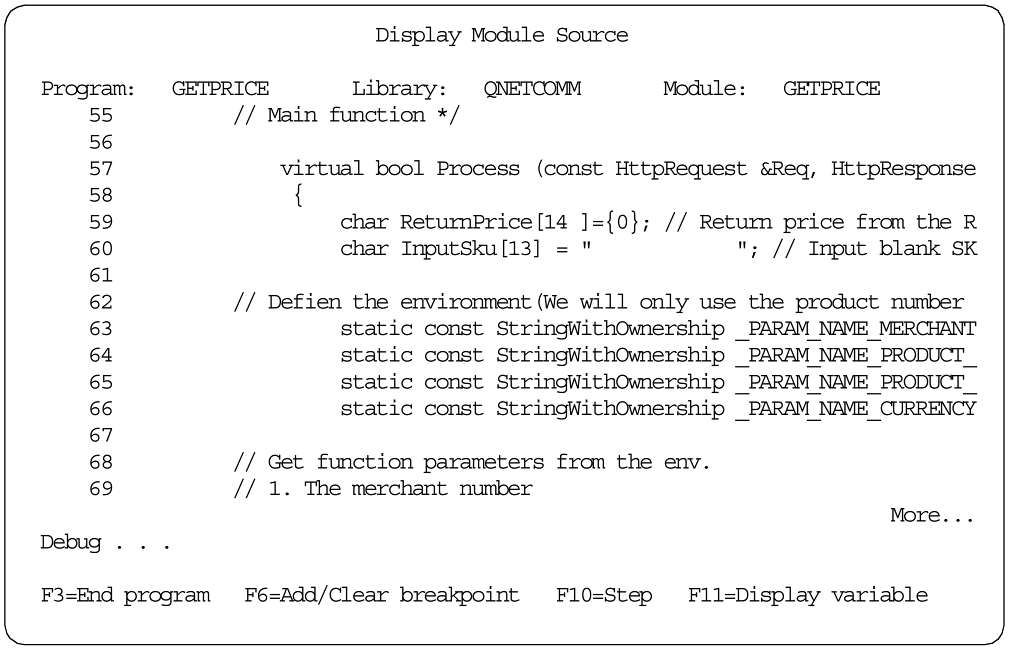


```
                             Display Module Source

 Program:   GETPRICE       Library:   QNETCOMM       Module:   GETPRICE
               //**********************************************************/
               // This function will call the back end system api in order */
               // to calculate an item price.                              */
               //                                                          */
               //                                                          */
               //**********************************************************/

               #ifdef AS400
                       #include "coibm.h"
               #endif

               #include <bcd.h>     // For working with packed decimal fields
               #include "objects/objects.pch" // Net.Commerce include

               // The following define is for compatibility with other platfo
                                                                         More...
 Debug . . .

 F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable
```

Figure 426. Source Debug Screen

4. Go to your browser and press on the link to display product information. The product display command calls the GET_BASE_UNIT_PRICE task. The task calls our overridable function.

   Return to your session. The Net.Commerce server job should stop and display the first statement of your source code as shown in Figure 427 on page 441.



```
                           Display Module Source

Program:   GETPRICE       Library:   QNETCOMM       Module:   GETPRICE
                 // Main function */

                     virtual bool Process (const HttpRequest &Req, HttpResponse
                      {
                          char ReturnPrice[14 ]={0}; // Return price from the R
                          char InputSku[13] = "            "; // Input blank SK

                 // Defien the environment(We will only use the product number
                          static const StringWithOwnership _PARAM_NAME_MERCHANT
                          static const StringWithOwnership _PARAM_NAME_PRODUCT_
                          static const StringWithOwnership _PARAM_NAME_PRODUCT_
                          static const StringWithOwnership _PARAM_NAME_CURRENCY

                 // Get function parameters from the env.
                 // 1. The merchant number
                                                                        More...
Debug . . .

F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable
```

Figure 427. Net.Commerce Source Breakpoint

You can now debug the program using standard AS/400 debugging capabilities.

19.6 Working with the Back-End System on a Different Server

In our example, the Net.Commerce site and the back-end system operated on the same AS/400 server. However, in many cases, your site will be installed on one server and your back-end system on another server. Luckily, the AS/400 system allows you to easily connect to and access remote databases. Describing the details of the different methods is beyond the scope of this book. Basically, we have two good options to connect with the remote back-end server:

- To perform native database operations, similar to the ones performed in the GETPRICER program, we can use DDM support.
- To perform SQL requests or embed SQL, we can use DRDA support.

Chapter 20. Writing Commands

To implement our site design, we did not need to write any new commands. As described in Chapter 19, “Implementing Overridable Functions” on page 413, commands represent a business process. Overridable functions represent precise pieces of business logic (for example, getting the product price). However, it is possible that your specific site design may require writing new commands.

This chapter guides you step-by-step through the process of writing a simple new command for Net.Commerce. This command does nothing but writes text to the user browser. The intent of this simple exercise is to help you become acquainted with the process involved in writing a completely new Net.Commerce command.

Note

To write commands, you must learn the low-level details of programming commands as described in the book *Commands, Tasks, Overridable Functions, and the E-commerce Framework.* This document is available in softcopy format only from the Net.Commerce Web site. Refer to E.4, “Other Resources” on page 536, for information about obtaining this document.

20.1 Creating Your Working Directory

Keep all files needed for command creation in one directory. The command we are about to write is called *MyNewCmd*. It makes sense to name your directory with that name. You can place your directory everywhere that you want it. However, we recommend that you have all of your directories for user-written commands under the same parent directory and on the root IFS directory. The CL code shown in Figure 428 creates the directory tree for our example.

```
PGM

 MD '/UserStuff'
 MD '/UserStuff/NetCommerce'
 MD '/UserStuff/NetCommerce/Commands'
 MD '/UserStuff/NetCommerce/Commands/MyNewCmd'

 ENDPGM
```

Figure 428. Creating the Directory Tree (Member CRTDIR in File QCLSRC)

Figure 429 displays the directory structure created by the CL program.

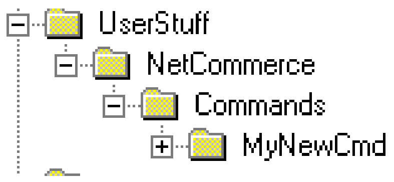


Figure 429. Directory Tree for the Example

20.2 Creating the Binding Directory

Binding directories are used by the AS/400 ILE environment to easily link modules with their relevant service programs and other modules. In our example, the binding directory is used to create the command service program. It includes all of the service programs that are needed to create the Net.Commerce NET command.

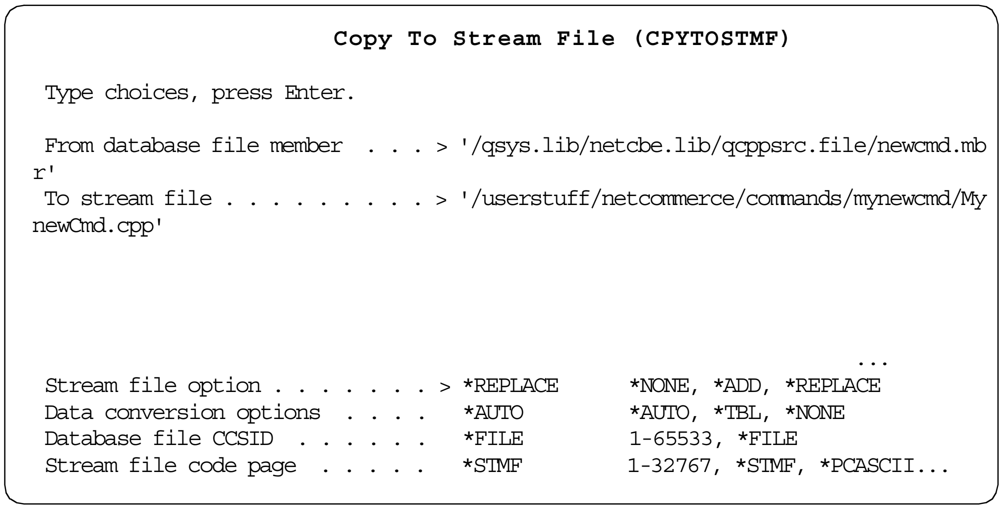
The CL program shown in Figure 430 creates the correct binding directory.

```
PGM

             CRTBNDDIR  BNDDIR(QGPL/NETCOMB) TEXT('Binding directory +
                          for Commands and OF')

             ADDBNDDIRE BNDDIR(QGPL/NETCOMB) +
                          OBJ((QNETCOMM/QNECONTAIN) +
                          (QNETCOMM/QNEMESSAGE) +
                          (QNETCOMM/QNECOMMON) (QNETCOMM/QNEDBC) +
                          (QNETCOMM/QNESVROBJ) +
                          (QNETCOMM/QNEPAYOBJS) (QNETCOMM/QNEPAYMSG))

 ENDPGM
```

Figure 430. Creating the Binding Directory (Member BNDDIRC in File QCLSRC)

20.3 Preparing the Source File

Our example is located in the QCPPSRC file and is called *newcmd*. Copy the new command example to the IFS root file system as shown in Figure 431.



```
                          Copy To Stream File (CPYTOSTMF)

 Type choices, press Enter.

 From database file member  . . . > '/qsys.lib/netcbe.lib/qcppsrc.file/newcmd.mb
r'
 To stream file . . . . . . . . . > '/userstuff/netcommerce/commands/mynewcmd/My
newCmd.cpp'




                                                                           ...
 Stream file option . . . . . . . > *REPLACE      *NONE, *ADD, *REPLACE
 Data conversion options  . . . .   *AUTO         *AUTO, *TBL, *NONE
 Database file CCSID  . . . . . .   *FILE         1-65533, *FILE
 Stream file code page  . . . . .   *STMF         1-32767, *STMF, *PCASCII...
```

Figure 431. Copy the Source File to the Root File System

The sample new command source is almost empty. It contains all of the function definitions for the Net.Commerce command, but has no logic in it. The command must be written in C++. Refer to Chapter 19, “Implementing Overridable

Functions" on page 413, for details on integrating RPG and other ILE and OPM language programs to C++. Figure 432 shows the sample new command source.

```
// This command source file is used to demonstrate writing new commands for Net.Commerc


        #include "objects/objects.pch"    // include Net.Commerce

        #if defined(WIN32)
                 #define __DLL_EXPORT__ __declspec(dllexport)
        #else
                 #define __DLL_EXPORT__
        #endif

        class __DLL_EXPORT__ NewCmd    : public NC_Command
        {
        public:
                NewCmd(void)
                {
                }
                virtual bool Initialize(void)  // Command init
                {
                    return true;
                }

                virtual  ~NewCmd(void)
                {
                }

                void operator delete( void* p ) { ::delete p; }

                virtual NC_Command* Clone(void) { return NULL; }
                public:
                virtual void FailedRegistration(NC_RegistrationID& RegID, const ErrorMsg_Reg* Err)
                {
                   error << indent << "Error: Registration failed           " << endl;
                }

                virtual bool CheckParameters (const HttpRequest& Req, HttpResponse& Res,
                NC_Environment& Env, NC_Environment& Resources)
                {
                    return true;
                }

                virtual bool FailedAccesControl(void)
                {
                   error << indent << "Error: Access failed                 " << endl;
                   return true;
                }

                virtual bool Process (const HttpRequest& Req, HttpResponse& Res,
                NC_Environment& Env)
                {
                     Res.setDocument("My First command worked !!");
                      return true;
                }
                };

                static bool X2 = NC_CommandManager::GetUniqueInstance().RegisterCommand("IBM", "NC",
                                "NewCmd", 1.0, new NewCmd);  // Register
```

Figure 432. The New Command Source

A detailed explanation of writing commands is not in the scope of this book. If you need to write new commands in your site implementation, refer to the document *Commands, Tasks and OFs and the E-commerce Framework*. This command simply writes text specified in the value of the Res.setDocument function to the user browser.

Use a PC editor or the AS/400 EDTF command to update the command source as prescribed here:

1. Replace the string NewCmd with the name of your new command (MyNewCmd).
2. Replace the Res.setDocument text value with some text that you will recognize as coming from your command, for example, `Res.setDocument("This is my test text")`.
3. Save the file.

20.4 Building the Command

To build the command, we have to create a service program with the command module in it. Here, we use the native C++ compiler because we do not have a need for any visual components. Refer to 2.2.2, “Optional AS/400 Net.Commerce Software Requirements” on page 11, for a list of software that allows you to compile C++ on the AS/400 system.

The CL program shown in Figure 433 creates the service program.

Note

Net.Commerce documentation refers to AS/400 service programs as DLLs.

```
PGM
      CRTCPPMOD  MODULE(MYLIB/TEMPMOD) +

SRCSTMF('/userstuff/netcommerce/commands/MyNewCmd/Mynewcmd.cpp') +
          OUTPUT('/userstuff/netcommerce/commands/mynewcmd/mynewcmd.lst')
+
           DEFINE(AS400) +
          INCDIR('/Qibm/proddata/netcommerce/adt/include')

      CRTSRVPGM  SRVPGM(MYLIB/MYNEWCMD) +
               MODULE(MYLIB/TEMPMOD) EXPORT(*ALL) +
               TEXT('Sample new Net.Commerce command') +
                BNDDIR(QGPL/NETCOMB)
 ENDPGM
```

Figure 433. Create Command Service Program (Member BLDCMD in File QCLSRC)

If any errors were encountered during compilation, you have to find the errors on the output file `/userstuff/netcommerce/commands/mynewcmd/mynewcmd.lst`. Correct the errors and recompile.

Note that we created the module in library MYLIB and without any debug information. You may need to compile more complicated commands with the debug option turned on as described in 19.5.1, “Compiling the Overridable Function with Debug Information” on page 437.

Also note that the service program is created in the user library MYLIB, rather than the QNETCOMM library. You should *never* add objects to the QNETCOMM library. If you do, your objects can become lost when you apply maintenance to Net.Commerce.

20.5 Registering the Command in the Database

There are several tables in the Net.Commerce database that deal with commands, tasks, and overridable functions. For our command to run, we must register it in the Net.Commerce database.

Note

For more information about writing stored procedures on the AS/400, refer to the redbook *DB2/400 Advanced Database Functions*, SG24-4249.

The following source member is used to create a SQL stored procedure that registers the command to the Net.Commerce database using these values:

- The command service program name (DLL) is `MYNEWCMD`.
- The instance name is `work`.
- The command name and URL are both `MyNewCmd`. It is accessible to the public through a URL.
- The command can run in the ncommerce pool.
- The command cannot run in SSL mode.

```
--***************************************************************/
-- This procedure is used as an example to register a new command */
--                                                              */
--   Author: Shahar mor                                         */
--   Provided AS IS                                             */
--***************************************************************/
CREATE PROCEDURE NETCBE.REGCMD
LANGUAGE SQL MODIFIES SQL DATA
BEGIN
DECLARE pool_nbr int;
DECLARE cmd_nbr int;
DECLARE acc_mod int;

insert into work.cmds (refnum, dll_name,vendor,product,name,version,
                 url,export,description)
       select max(refnum) + 1, 'MYLIB/MYNEWCMD', 'MyCompany' ,
              'MyProduct', 'MyNewCmd',1.0,'MyNewCmd',1,
              'My First New Command' from work.cmds;
Select  refnum into pool_nbr from work.pools where name= 'ncommerce'
Select  refnum into cmd_nbr from work.cmds where vendor= 'MyCompany'
and product = 'MyProduct' and name = 'MyNewCmd' and version = 1.0;
insert into work.pool_cmd (pool_rn, cmd_rn) values(pool_nbr, cmd_nbr);

Select  max(refnum) + 1 into acc_mod from work.acc_mode;
insert into work.acc_mode (refnum, cmd_refnum, ssl,protect)
values(acc_mod,cmd_nbr,0,0);

commit;
END
```

Figure 434. Registering the Command (Member REGCMD in File QSQLSRC)

After you type the SQL procedure shown in Figure 434, you must compile it. To compile the source into a stored procedure, type the following information on an AS/400 command line and press **Enter**:

```
RUNSQLSTM SRCFILE(NETCBE/QSQLSRC)
          SRCMBR(REGCMD)
          NAMING(*SQL)
          OUTPUT(*PRINT)
```

This creates a stored procedure that is called from SQL. To execute the stored procedure, type the following command on an AS/400 command line:

```
STRSQL
```

Press **Enter**. This starts the interactive SQL. Type the following command on the SQL command line:

```
CALL REGCMD
```

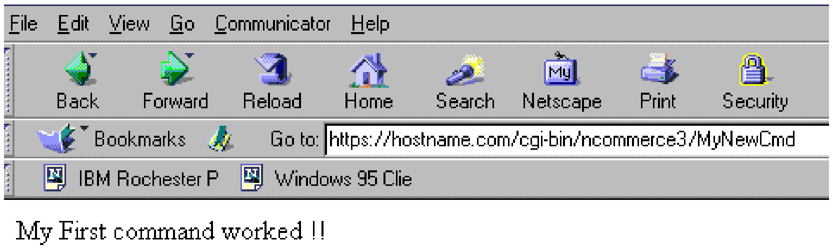
Press **Enter**. This updates the CMDS, POOL_CMD, and ACC_MODE tables in the library specified in the stored procedure. In our sample, this was the collection WORK.

For the new command to take affect, restart the Net.Commerce server after the command registration completed.

20.6 Testing the New Command

You can easily test the new command from a browser by directly specifying the URL: `http://<hostname>/cgi-bin/ncommerce3/MyNewCmd`.

In this URL, the host name is the complete host name of the server. You should see an HTML page with your text on it similar to the one shown in Figure 435.


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks
Go to:
IBM Rochester P
Windows 95 Clie

https://hostname.com/cgi-bin/ncommerce3/MyNewCmd

My First command worked !!

Figure 435. Output of Our First Command

Note

Net.Commerce commands are case sensitive.

More advanced commands are, of course, more difficult to test. For example, if you perform updates on the database, you must check manually whether your command changed the database correctly.

If Net.Commerce does not load or run your command, look in the normal log files of Net.Commerce described in 21.1, "Net.Commerce Server Logs" on page 453, for a description of the error. You can also write to the log files from your command to help catch errors.

20.7 Coding Patterns and Guidelines

When writing new commands and overridable functions, you code the same kinds of functions again and again. Some of these functions are described here with short code listings so that you can reuse them on a regular basis. In addition, some general coding considerations and guidelines are addressed here. Our examples are all in the C++ language. See Chapter 19, "Implementing Overridable Functions" on page 413, for an example that calls an RPG program from C++.

20.7.1 Calling a Task

The following code fragment executes a task called taskName:

```
const ErrorMsg_Cmd* Err;
Err = NC_OverridableFunctionManager::Call(Req, Res, Env, "taskName",
MerchantRefNum);
if (Err == &_ERR_CMD_ERR_HANDLED)
throw Err;
if (Err != NULL) // Otherwise, just return a failure.
return false;
```

If successful, NULL is returned in the Err variable. Possible error codes are:

- `&_ERR_CMD_ERR_HANDLED` — The OF generated its own error page
- `&_ERR_CMD_BAD_PROCESS_API` — Error from the OFs process method
- `&_ERR_CMD_API_NOT_FOUND` — OF not found

If the returned error is `&_ERR_CMD_ERR_HANDLED`, you should throw the Err object, which returns the error page that was generated by the OF. In the other cases, you should return "false".

For a more detailed approach to error handling, refer to book *Commands, Tasks, Overridable Functions, and the E-commerce Framework*. This document is available in softcopy format only from the Net.Commerce Web site. Refer to E.4, “Other Resources” on page 536, for information about obtaining this document.

20.7.2 Using Iterators

This code fragment shows how to declare an iterator that is used in a loop to find all parameters of a command or an OF:

```
NameValuePairMap::Iterator I(&req.getNVPs());
for (I.Begin(); *I != NULL; ++I)
{
trace<<(*I)->getName()<<endl;
trace<<(*I)->getValue()<<endl;
}
```

Note the use of `I.Begin()` to find the first name-value pair and the use of the ++ operator to find the next pair until the value of the iterator becomes NULL.

20.7.3 Selecting Rows from the Database

Important

We *highly* recommend that you do not use native database support or Net.Data macros to update the Net.Commerce collection. However, you can update your back-end system tables using native database support.

This code fragment shows you how to select rows from the database:

```
int returnCode;
String Stmt;
Stmt.Resize(STRLEN_1K);
DataBase& DB = (DataBase&)*Misc::CheckEnvironmentVariable(Env,
NC_Environment::_VAR_MainDatabase); // get the data base object
if(DB==NULL)
return false;
Stmt << "select stpcode from shipto " << endl
<< "where stornbr=" << OrderRefNum << " and stsanbr=" << AddressRefNum;
SQL Sql(*DB,Stmt);
for(returnCode=Sql.getNextRow(SqlRow);returnCode ==
ERR_DB_NO_ERROR;returnCode=Sql.getNextRow(SqlRow))
{
SqlRow.getCol(1).getVal(stpcode);
}
if (returnCode != ERR_DB_NO_ERROR && returnCode != ERR_DB_NO_DATA)
{
Sql.ReportError ();
return false;
```

20.7.4 Updating Rows in the Net.Commerce Database

Important

We *highly* recommend that you do not use native database support or Net.Data macros to update the Net.Commerce collection. However, you can update your back-end system tables using native database support.

This code fragment shows you how to update rows in the Net.Commerce database:

```
int returnCode;
String Statement;
Statement.Resize(256);
Statement.Clean();
DataBase& DB = (DataBase&)*Misc::CheckEnvironmentVariable(Env,
NC_Environment::_VAR_MainDatabase);
if(DB==NULL)
return false;
Statement << "UPDATE SHOPPINGS SET SBFIELD1 = " << endl
<< quantity << endl
<< " WHERE SBSHNBR = " << shopperRefNum << endl
<< " AND SBMENBR = " << merchantRefNum << endl
<< " AND SBPRNBR = " << productRefNum;
SQL sql(*DB,Statement);
returnCode=sql.getSQLrc();
if(returnCode != ERR_DB_NO_ERROR)
{
sql.ReportError ();
return false;
```

The Net.Commerce server command controls the commit scope. If you wish to update tables in the Net.Commerce collection, do it by using the database classes.

20.7.5 Static Variables

Although Net.Commerce is not yet a multi-threaded environment, it is a good idea to make your code thread safe now. One precaution is that you should not use static variables unless they are declared as constant. The only exception to this rule is when registering the command or OF.

Important

Do not make any assumptions with respect to the threaded nature of the environment. We intend to make the servers multi-threaded. In this case, your commands and OFs may, one day, have to run in a multi-threaded environment. Avoid using non-constant static objects to store state information.

20.7.6 Security Considerations

Keep in mind that end users can see the URL that makes up the HttpRequest when a command is called. Therefore, they will know such aspects as the name of the command, the name of the parameters, and the values of the parameters. Some people can change the parameter values to get your store to behave differently.

Never assume the correctness of the passed parameters in your command. You should implement the check parameters function in your command. Your command should always have proper access control defined for it, and guard against common attacks.

The data you receive should be checked against each other. For example, your command could get a product number and a merchant number, and assume that the product belongs to the merchant. *Do not assume. Be sure.* When you look up the product from the database, simply constrain the query with the merchant ID. Also if the two parameters do not agree, you can simply return "false." Either the calling page was not coded properly or someone is changing your site, which means it is OK to return the system error page.

Chapter 21. Site Administration

Administration is divided into *technical administration* and *content administration*. Technical administration refers to all components needed to set up the system and keep it a live. Content administration is responsible for products, categories, shoppers management, and log analyses.

21.1 Net.Commerce Server Logs

Net.Commerce provides logging mechanisms to allow users to track various activities related to their site. In particular, two main types of logging features are available: system logs and user traffic logs.

21.1.1 System Log

System logs basically track every action taken by the system when the server is accessed. These logs are especially helpful when debugging code. The directory path where these logs are located is associated with the MS_LOGPATH variable in the Net.Commerce configuration file. When this directory is examined, you will find that there are two types of system log files.

The first type starts with the word "control" followed by the date and some other numbers. This file traces various routines that must take place when the Net.Commerce server first starts up. Some of these routines include:

- Opening of database connection
- Preparing the server for the default command pool
- Preparing the default command receptors and OF receptors
- Registering the default commands and OFs
- Initializing the default commands and OFs

This log is useful only if something goes wrong during the server's initialization. In these cases, the log file can be scanned for any failures, unexpected conditions, or missing items.

The second type of system log file starts with the word "ncommerce" followed by the date and some other numbers. Initially, it traces the exact same routines as described above except that the ncommerce command pool is used instead of the default command pool. More importantly, it proceeds to document every event that occurs on the site. For every transaction that occurs (such as adding an item to the shopping cart), every command, task, and OF called to perform this transaction is listed in order of execution. In addition, each transaction has three types of messages that are possibly output:

- **STATUS** — Message to output general information such as which command or OF is currently being executed and whether it was successful.
- **DEBUG** — Message to output specific lines of code that were specified by the programmer for debugging purposes.
- **ERROR** — Message to output any processing errors that may have occurred.

Finally, if applicable, each transaction section ends by displaying various environment parameters as well as the actual HTML that is returned to the browser when the transaction completes. You may discover that there are multiple ncommerce-type log files in the log directory. This may occur for various reasons. For example, every time the server is stopped and restarted, a new log file is created. For whatever the reason, you may find that although every transaction is logged, sometimes you have to search through more than one file to find just the right one. Your best option is to start with the most current ones. Another suggestion is to constantly maintain your log directory by deleting old log files. Because they track every system event that occurs, the log files can grow in size very quickly and take up valuable hard drive space.

21.1.2 User Traffic Log

The user traffic log is used to track the activity of users accessing your site. It is useful to determine the browsing and buying patterns of your shoppers, possibly to address personalization or marketing issues. Unlike the system logs, the user traffic log is maintained in the USRTRAFFIC database table instead of a text file. It currently consists of 16 fields, some of which are discussed in the next section. Some of the user traffic data includes the visitor login time and date, the IP address of the visitor, and the specific query string used to display visited pages. Every single process that is executed on the site is saved as a full record in the table.

By default, the user traffic log is turned off. To turn it on, change the value of the USERTRAFFIC_LOG variable in the Net.Commerce configuration file from "0" to "1" (USERTRAFFIC_LOG 1). To turn the logging off again, either change the value of the variable back to "0," or remove the variable and its value entirely.

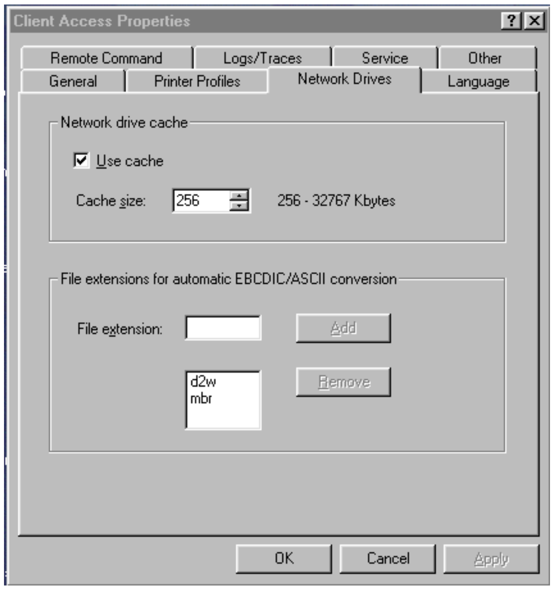
21.1.3 Viewing the Log Files

Since the user traffic log is kept in the Net.Commerce database table USRTRAFFIC, you can view the table's contents by using a standard AS/400 file browsing tool such as STRSQL.

However, the server logs are kept as stream files in the IFS root directory and cannot be displayed by database requests. You can use one of the following methods to display the contents of these log files:

- Use the `EDTF` command from an AS/400 emulation screen as described in 7.3, "Stream File Handling Tools" on page 111. The problem with EDTF is that the log file name is very long and it is not convenient to type it in the EDTF prompt.
- Map the network drive using Client Access support and use PC editor such as Wordpad to browse the log file contents. The log files are in EBCDIC so you must also instruct Client Access to perform the translation from EBCDIC to ASCII. Complete the following steps to configure Client Access automatic log file translations:
  1. From the Client Access folder, click on **Client Access properties**.
  2. In the Client Access properties window, click on the **Network Drives** tab.

     You see the window that appears in Figure 436 on page 455.



Network drive cache

[✓] Use cache

Cache size: 256 256 - 32767 Kbytes

File extensions for automatic EBCDIC/ASCII conversion

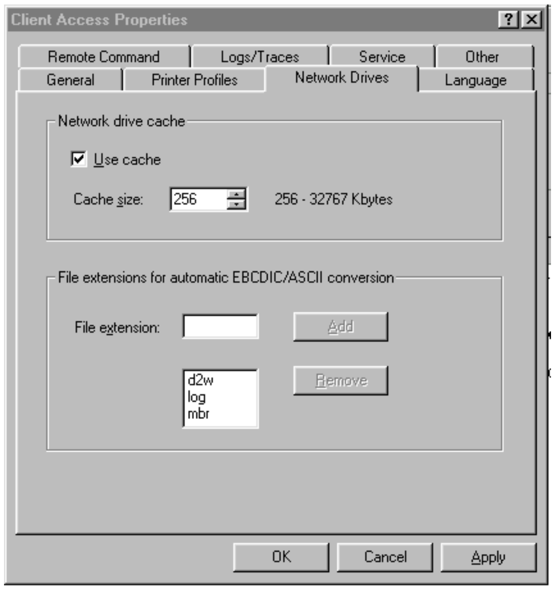
File extension:

d2w
mbr

Figure 436. Define Network Drive Attributes

3. In the File extension field, type `log`, and click on the **Add** button.

   You should see the log extension in the bottom of the window as shown in Figure 437.



Network drive cache

[✓] Use cache

Cache size: 256 256 - 32767 Kbytes

File extensions for automatic EBCDIC/ASCII conversion

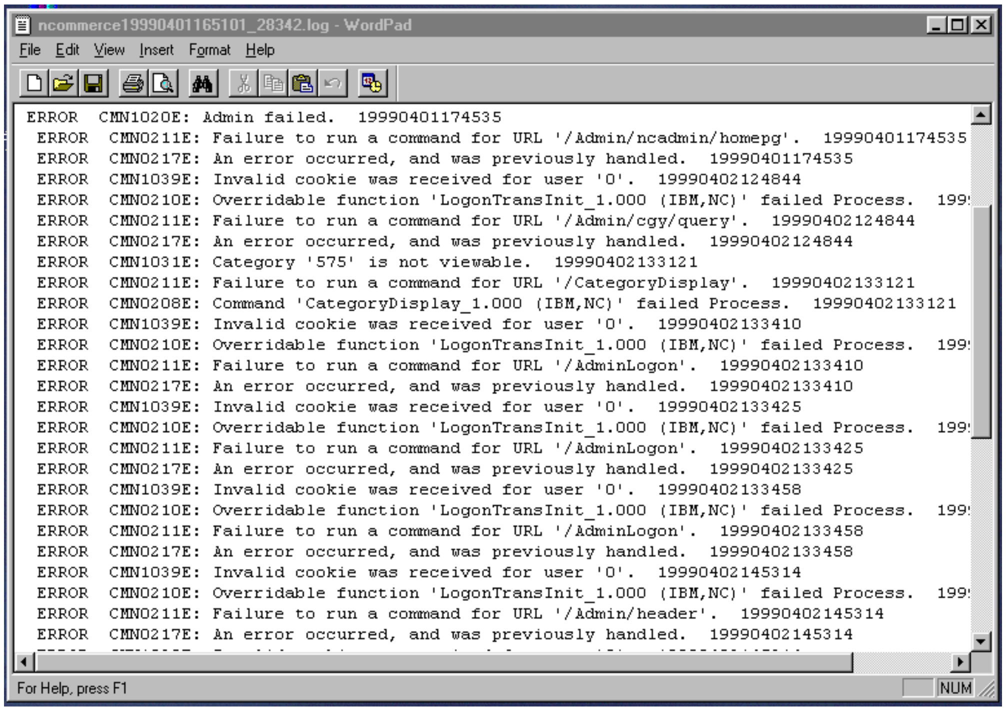
File extension:

d2w
log
mbr

Figure 437. Define Automatic Conversion for Log Extension Complete

Client Access takes care of EBCDIC to ASCII translation for any file with the log extension.

You can now browse the file contents from a PC editor. Figure 438 on page 456 shows an example log file. The beginning of the file name is ncommerce19990401and reflects the log date. The log file is located in directory /Qibm/UserData/NetCommerce/Instance/Instancename/logs


ncommerce19990401165101_28342.log - WordPad
File Edit View Insert Format Help
For Help, press F1
NUM

```
ERROR  CMN1020E: Admin failed.  19990401174535
 ERROR  CMN0211E: Failure to run a command for URL '/Admin/ncadmin/homepg'.  19990401174535
 ERROR  CMN0217E: An error occurred, and was previously handled.  19990401174535
 ERROR  CMN1039E: Invalid cookie was received for user '0'.  19990402124844
 ERROR  CMN0210E: Overridable function 'LogonTransInit_1.000 (IBM,NC)' failed Process.  199
 ERROR  CMN0211E: Failure to run a command for URL '/Admin/cgy/query'.  19990402124844
 ERROR  CMN0217E: An error occurred, and was previously handled.  19990402124844
 ERROR  CMN1031E: Category '575' is not viewable.  19990402133121
 ERROR  CMN0211E: Failure to run a command for URL '/CategoryDisplay'.  19990402133121
 ERROR  CMN0208E: Command 'CategoryDisplay_1.000 (IBM,NC)' failed Process.  19990402133121
 ERROR  CMN1039E: Invalid cookie was received for user '0'.  19990402133410
 ERROR  CMN0210E: Overridable function 'LogonTransInit_1.000 (IBM,NC)' failed Process.  199
 ERROR  CMN0211E: Failure to run a command for URL '/AdminLogon'.  19990402133410
 ERROR  CMN0217E: An error occurred, and was previously handled.  19990402133410
 ERROR  CMN1039E: Invalid cookie was received for user '0'.  19990402133425
 ERROR  CMN0210E: Overridable function 'LogonTransInit_1.000 (IBM,NC)' failed Process.  199
 ERROR  CMN0211E: Failure to run a command for URL '/AdminLogon'.  19990402133425
 ERROR  CMN0217E: An error occurred, and was previously handled.  19990402133425
 ERROR  CMN1039E: Invalid cookie was received for user '0'.  19990402133458
 ERROR  CMN0210E: Overridable function 'LogonTransInit_1.000 (IBM,NC)' failed Process.  199
 ERROR  CMN0211E: Failure to run a command for URL '/AdminLogon'.  19990402133458
 ERROR  CMN0217E: An error occurred, and was previously handled.  19990402133458
 ERROR  CMN1039E: Invalid cookie was received for user '0'.  19990402145314
 ERROR  CMN0210E: Overridable function 'LogonTransInit_1.000 (IBM,NC)' failed Process.  199
 ERROR  CMN0211E: Failure to run a command for URL '/Admin/header'.  19990402145314
 ERROR  CMN0217E: An error occurred, and was previously handled.  19990402145314
```

Figure 438. Net.Commerce Log File Example

21.2 Database Cleanup Utility

The Net.Commerce Database Cleanup utility allows you to delete a number of unnecessary records at the same time. You can delete the following record types:

- Guest shoppers
- Temporary shopper addresses
- Old orders
- Products that are marked for deletion
- User-traffic log records
- Records in the STAGLOG table that were propagated to the production database
- Records in the CACHLOG table that identify invalid cache pages that were purged
- Stores

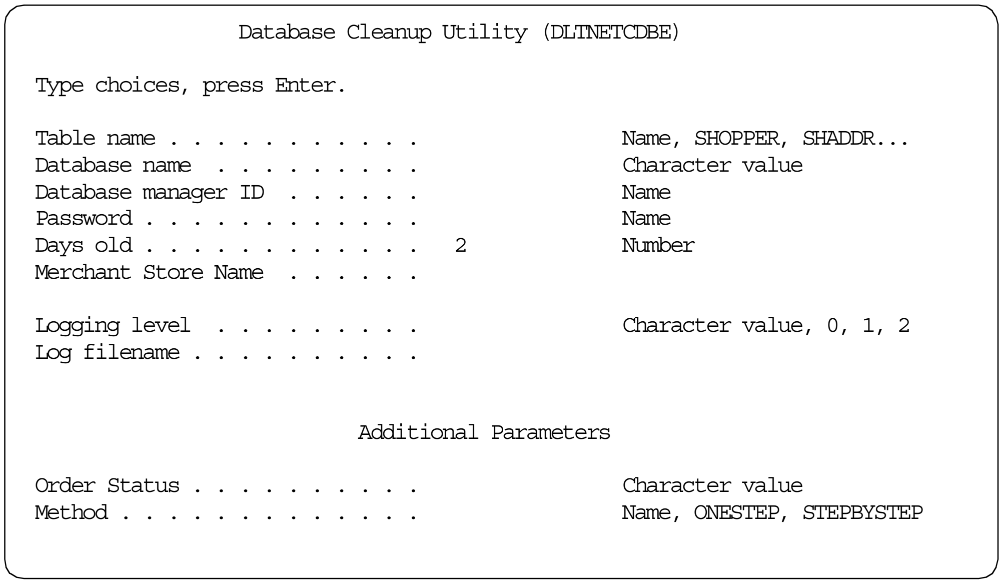
When the Database Cleanup utility deletes a record in a table, it also deletes the corresponding records in other tables that are linked to that table, to preserve the referential integrity of the database. By default, the Database Cleanup utility writes to a log file to the file:

`/QIBM/UserData/NetCommerce/Instance/<instance_name>/Logs/dbclog.txt`

Here, `<instance_name>` is the name of the Net.Commerce instance.

The administrator that runs this utility must sign on as the instance name for the instance whose records are being deleted. Or they must add the instance library to their session library list before calling the cleanup utility.

To use the clean database utility, type the command `DLTNETCDBE`, and press **F4.** The command prompt appears as shown in Figure 439.



```
                 Database Cleanup Utility (DLTNETCDBE)

 Type choices, press Enter.

 Table name . . . . . . . . . . .                Name, SHOPPER, SHADDR...
 Database name  . . . . . . . . .                Character value
 Database manager ID  . . . . . .                Name
 Password . . . . . . . . . . . .                Name
 Days old . . . . . . . . . . . .   2            Number
 Merchant Store Name  . . . . . .

 Logging level  . . . . . . . . .                Character value, 0, 1, 2
 Log filename . . . . . . . . . .


                          Additional Parameters

 Order Status . . . . . . . . . .                Character value
 Method . . . . . . . . . . . . .                Name, ONESTEP, STEPBYSTEP
```

Figure 439. The DLTNETCDBE Command Prompt

The command parameter description is shown in Table 24.

Table 24. DLTNETCDBE Command Prompt

Parameter	**Parameter Description**
Table Name (TBLNAME)	The name of the table from which we want to clean records.
Data base name (DATABASE)	The name of the database where the Net.Commerce tables are located. Use the `WRKRDBDIRE` command to find this name.
Data base manager ID (DBUSR)	Specifies the logon ID of the administrator who has been assigned access to the database. If this parameter is not specified, the ID of the user invoking the utility is used. In this case, make sure you have the proper authority to the Net.Commerce instance tables.
Password (DBPASWD)	Specifies the password of the user whose logon ID was specified in the DBUSR parameter. If the command is issued on the local Net.Commerce site and did not specify user name on the DBUSR parameter, you can omit this parameter.

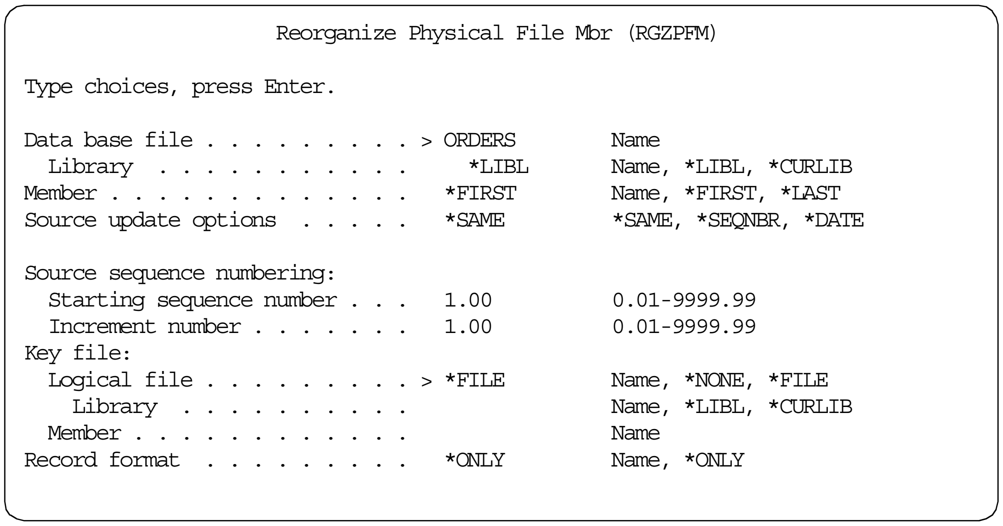
Parameter	Parameter Description
Days Old (DAYSOLD)	Specifies the minimum age, in days, of records to be deleted. The default is two days. If this parameter is not specified, all records that are more than two days old are deleted. To determine the age of the record, the utility compares the current date and time with the date and time in the timestamp column of the table.
Merchant store name (STORENAME)	The name of the store to be deleted. Since the store name is case sensitive, put the store name in single quotation marks ('). The utility deletes records associated with the store from all tables checked by the Database Cleanup utility.
Logging level (LOGLEVEL)	Specifies the level of logging to be performed during the database clean-up: 0 - Specifies that no logging is to be performed. 1 - Specifies that log records are to be created only for deletions from the table specified. 2 - Specifies that log records are to be created for deletions of records from the table specified, and for any deletions of subordinate records from other tables. On normal terms, you can use the log level of 1.
Log File (LOGFILE)	Specifies the path and name of the log file. If this parameter is not specified, the log file dbclog.txt is created in the instance root directory (/QIBM/UserData/NetCommerce/Instance/<instance _name>/Logs/). The issuer of the command must have write authority to the specified path, and the path must already exist. Do not direct the log output to the QSYS.LIB file system.
Order status (ORDSTAT)	If the ORDERS table is specified, it indicates the status of orders to be deleted. This parameter corresponds to the content of column ORSTAT. C - Orders are in a complete state. X - Orders are in a cancelled state. P - Orders are in a pending state. U - Orders are in a user-defined state.
Method (METHOD)	The commit method for deleting the store's records. This is an optional parameter. If this parameter is not specified, the Instep method is used. **Instep** — Commits only once after all records are successfully deleted. If an error occurs during the deletion, the database rolls back to its original state. **StepByStep** — Commits each time a record is successfully deleted. If an error occurs while a store's records are deleted, all of the previously committed deletions cannot be rolled back. Then, the store is left in a partially deleted state.

Figure 440 on page 459 displays an example to the log file created by the clean database utility. You can use the `EDTF` command described in 7.3, “Stream File Handling Tools” on page 111.

```
Mon Apr  5 11:24:46 1999
 CMN1501I Database Cleanup Utility started.
   19990405112447CMN1521I Row(s) from CACHLOG table deleted successfully.
   19990405112447[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
 [CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;

[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;
[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;


******
STATUS CMN0003S: Database 'AS01' has been commited.
******

[CACHLOG] xx; 100; 290; 1999-01-01-22.00.00.000000;


******
STATUS CMN0003S: Database 'AS01' has been committed.
******

  19990405112447

Mon Apr  5 11:24:47 1999 CMN1502I Database Cleanup Utility completed.
  19990405112447
======================================================================
```

Figure 440. Sample Cleanup Utility Log File

After the cleanup utility deletes the rows from the tables, you are advised to perform table reorganization to the tables processed by the utility. For example, Figure 441 on page 460 displays the command to reorganize the orders table after it was cleaned by the DLTNETCDBE command.



```
                    Reorganize Physical File Mbr (RGZPFM)

Type choices, press Enter.

Data base file . . . . . . . . . > ORDERS        Name
  Library  . . . . . . . . . . .     *LIBL       Name, *LIBL, *CURLIB
Member . . . . . . . . . . . . .   *FIRST        Name, *FIRST, *LAST
Source update options  . . . . .   *SAME         *SAME, *SEQNBR, *DATE

Source sequence numbering:
  Starting sequence number . . .   1.00          0.01-9999.99
  Increment number . . . . . . .   1.00          0.01-9999.99
Key file:
  Logical file . . . . . . . . . > *FILE         Name, *NONE, *FILE
    Library  . . . . . . . . . .                 Name, *LIBL, *CURLIB
  Member . . . . . . . . . . . .                 Name
Record format  . . . . . . . . .   *ONLY         Name, *ONLY
```

Figure 441. Reorganize the Net.Commerce Table after Cleanup Utility

For more information on using the cleanup utility, refer to the Net.Commerce online documentation.

21.3 Clearing Log Files

The Cleanup utility only deletes rows from the Net.Commerce database tables. However, the log files of Net.Commerce are located in the root file directory and are not cleaned by the database cleanup utility.

The following sample source code from a CL program can be used for saving and deleting Net.Commerce log files for a specific instance. Note that you cannot delete a log file that is currently used by the Net.Commerce server. You will have to perform the log cleanup when the Net.Commerce server is not active.

```
/**********************************************************/
/* This program is a simple example to let you clean      */
/* net commerce log files from the IFS.                   */
/*                                                        */
/* Arguments:                                             */
/*            pr_inst    - Instance                       */
/*                                                        */
/* Author: Shahar mor                                     */
/* Provided AS IS                                         */
/**********************************************************/
             PGM        PARM(&pr_inst)

             DCL        VAR(&pr_inst)    TYPE(*CHAR) LEN(10)

             DCL        VAR(&savstring)  TYPE(*CHAR) LEN(100)

/* General error variables */
             DCL        VAR(&msgflib)    TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgf)       TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgid)      TYPE(*CHAR) LEN(7)
             DCL        VAR(&msgdta)     TYPE(*CHAR) LEN(128)

             MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(STDERR))


/* Save the log files. Our example saves the files to a save file  */

             CHGVAR     VAR(&savstring) +
                          VALUE('/qibm/userdata/netcommerce/instance/+
                          ' *TCAT &pr_inst *TCAT '/logs/*.log')
             SAV        DEV('/qsys.lib/qgpl.lib/netcsavf.file') +
                          OBJ((&SAVSTRING)) SAVACT(*NO) CLEAR(*ALL)


/* Delete the log files                                            */

             DEL        +
                          OBJLNK('/qibm/userdata/netcommerce/instance+
                          /work/logs/*.log')

             RETURN
 STDERR:

             RCVMSG     MSGTYPE(*EXCP) MSGDTA(&msgdta) +
                                       MSGID(&msgid) MSGF(&msgf) +
                                       MSGFLIB(&msgflib)
             MONMSG     MSGID(CPF0000 MCH0000)
             SNDPGMMSG  MSGID(&msgid) MSGF(&msgflib/&msgf) +
                          MSGDTA(&msgdta) MSGTYPE(*ESCAPE)
             MONMSG     MSGID(CPF0000 MCH0000)

            RETURN

            ENDPGM
```

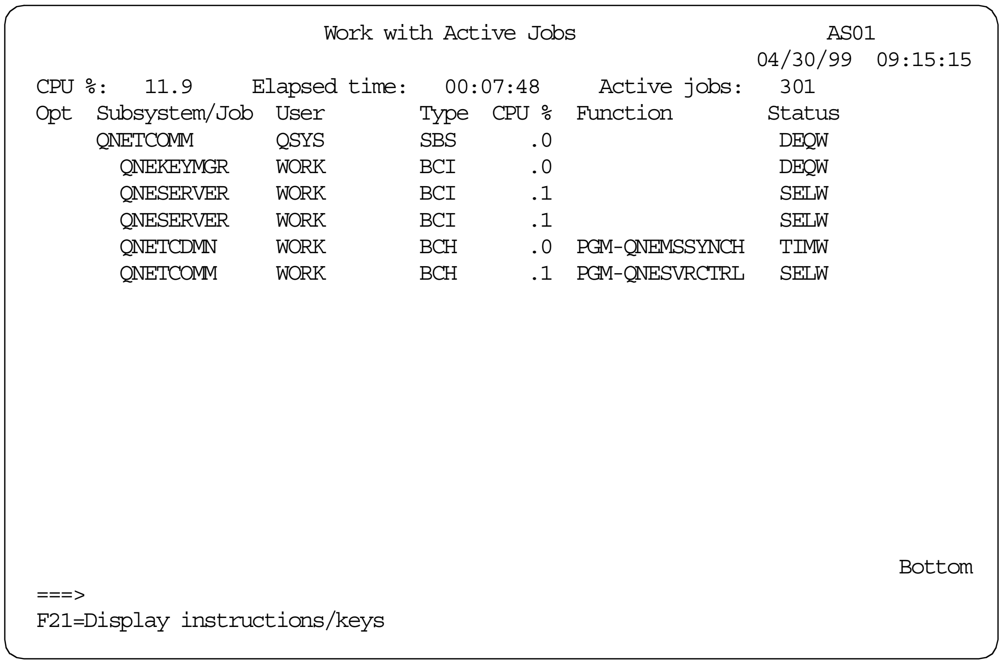
21.4 General Administration Tasks

Net.Commerce uses features in the OS/400 operating system. You must ensure that the AS/400 environment setup will allow Net.Commerce to function properly. In addition, the system administrator must perform the following tasks:

- Keep the AS/400 operating system and licensed program PTF levels current. To check the PTF level on your system, use the DSPPTF command. For more information, see *Basic System Operation, Administration, and Problem Handling Version 4,* SC41-5206.
- Make sure the TCP/IP services and the Net.Commerce HTTP server come up automatically after the system restarts. This can be done by including the STRTCP command in your startup program. For more information, see *TCP/IP Configuration and Reference Version 4*, SC41-5420.
- In case of a separated Net.Commerce and back-end application server, make sure that the communication link between the servers is up. You can check communication. For more information, see *Communications Management Version 4*, SC41-5406.
- Ensure that security exit programs connected to the AS400 JDBC server allows Net.Commerce traffic. The Net.Commerce product advisor uses JDBC connection to the Net.Commerce database. For more information, see *Tips and Tools for Securing Your AS/400 Version 4,* SC41-5300.

21.5 Net.Commerce Jobs on the AS/400 System

Net.Commerce runs on the AS/400 system in a subsystem called QNETCOMM. The jobs run with the user profile of your Net.Commerce instance. The server controller job QNETCOMM starts the QNESERVER daemons. The QNEKEYMGR handles the KEYS table.

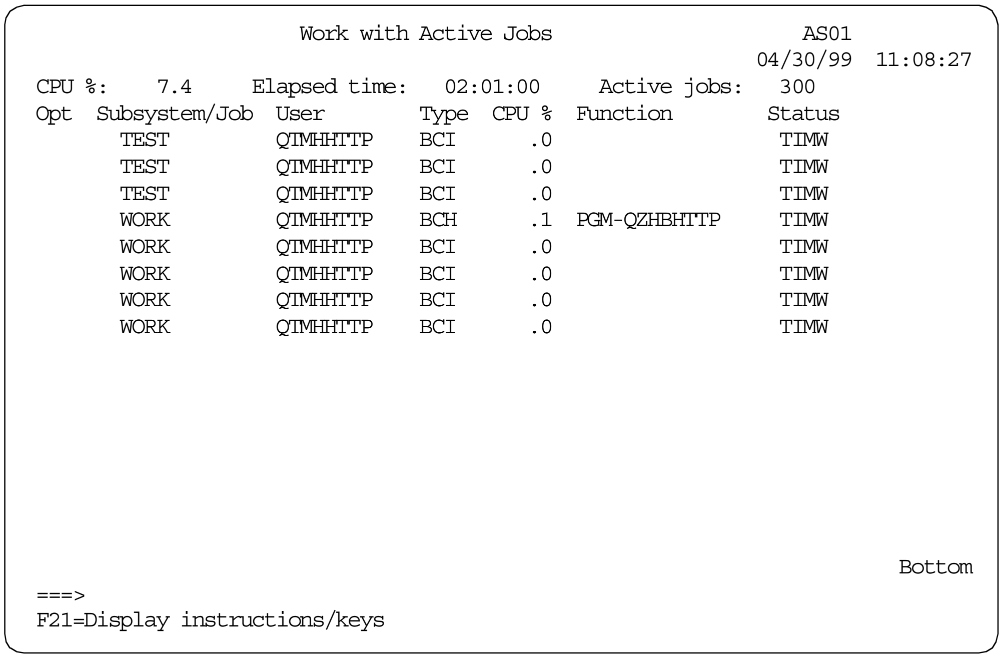


```
                         Work with Active Jobs                     AS01
                                                       04/30/99  09:15:15
CPU %:   11.9     Elapsed time:   00:07:48     Active jobs:   301
Opt  Subsystem/Job  User         Type  CPU %  Function        Status
     QNETCOMM       QSYS         SBS      .0                   DEQW
       QNEKEYMGR    WORK         BCI      .0                   DEQW
       QNESERVER    WORK         BCI      .1                   SELW
       QNESERVER    WORK         BCI      .1                   SELW
       QNETCDMN     WORK         BCH      .0  PGM-QNEMSSYNCH   TIMW
       QNETCOMM     WORK         BCH      .1  PGM-QNESVRCTRL   SELW

                                                                   Bottom
===>
F21=Display instructions/keys
```

Figure 442. Net.Commerce Subsystem

21.6 Web Server Jobs on the AS/400 System

The AS/400 HTTP Web Server runs in the subsystem QHTTPSVR.



```
                       Work with Active Jobs                     AS01
                                                       04/30/99  11:08:27
CPU %:    7.4     Elapsed time:   02:01:00     Active jobs:   300
Opt  Subsystem/Job  User         Type  CPU %  Function        Status
       TEST         QTMHHTTP     BCI      .0                   TIMW
       TEST         QTMHHTTP     BCI      .0                   TIMW
       TEST         QTMHHTTP     BCI      .0                   TIMW
       WORK         QTMHHTTP     BCH      .1  PGM-QZHBHTTP     TIMW
       WORK         QTMHHTTP     BCI      .0                   TIMW
       WORK         QTMHHTTP     BCI      .0                   TIMW
       WORK         QTMHHTTP     BCI      .0                   TIMW
       WORK         QTMHHTTP     BCI      .0                   TIMW

                                                                   Bottom
===>
F21=Display instructions/keys
```

Figure 443. HTTP Server Jobs

Appendix A. Source Code Samples

The sample code provided in this book is for the purpose of demonstrating how to tailor your site by modifying and extending Net.Commerce functionality. This code has not been fully tested for release into a production environment. These samples, as well as the samples shown throughout this redbook, are available from the ITSO Web site at: `http//www.redbooks.ibm.com`

From this site, select **Additional Materials** and locate the directory named **SG245198**.

A.1 Retrieving Encrypted Text

The following command and CL program returns the encrypted text for a given string. It can be used to add shoppers with their password from an existing back-end application to Net.Commerce.

```
/*****************************************************************/
/* Retrieve encrypted text command.                              */
/*                                                               */
/* Compile with ALLOW(*IPGM *BPGM)                               */
/* Author: Shahar Mor                                            */
/* Provided AS IS                                                */
/*****************************************************************/

             CMD        PROMPT('Retrieve Net.Commerce key')

             PARM       KWD(KETSTR) TYPE(*CHAR) LEN(10) MIN(1) +
                          PROMPT('Character to recieve key for')

             PARM       KWD(RTNKEY) TYPE(*CHAR) LEN(25) RTNVAL(*YES) +
                          PROMPT('Returned key')
             PARM       KWD(RTNHEX) TYPE(*CHAR) LEN(50) RTNVAL(*YES) +
                              PROMPT('Return hexadecimal key')
```

Figure 444. Command RTVENCKEY Source

```
/*********************************************************/
/* This program can be used to allow password sync. between*/
/* back end system customers and the Net.Commerce shoppers */
/* table.                                                  */
/*                                                         */
/* This program should be with public authority *exclude   */
/*                                                         */
/* Arguments:                                              */
/*            pr_string  - Original string                 */
/*            io_enc     - Return encrypted string         */
/*            io_enchex  - Return hexadecimal encrypted    */
/*                         string.                         */
/* Author: shahar mor                                      */
/* Provided AS IS                                                */
/*********************************************************/
             PGM        PARM(&pr_string  +
                             &io_enc     +
                             &io_enchex)

             DCL        VAR(&pr_string)  TYPE(*CHAR) LEN(10)
             DCL        VAR(&io_enc   )  TYPE(*CHAR) LEN(25)
             DCL        VAR(&io_enchex)  TYPE(*CHAR) LEN(50)

             DCL        VAR(&msgflib)    TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgf)       TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgid)      TYPE(*CHAR) LEN(7)
             DCL        VAR(&msgdta)     TYPE(*CHAR) LEN(128)

             DCL        VAR(&keyrow)     TYPE(*CHAR) LEN(23 )  +
                                         VALUE('Encrypted string (char)')
             DCL        VAR(&hexrow)     TYPE(*CHAR) LEN(23 )  +
                                         VALUE('Encrypted string (hex):')
             DCLF       FILE(QTXTSRC)

/* Create file for results        */

             DLTF       FILE(QTEMP/NETCSRC@)
             MONMSG     MSGID(CPF2105) /* Not found */

             CRTSRCPF   FILE(QTEMP/NETCSRC@)             +
                          MBR(NETCKEY) TEXT('Retrieve encrypted text')

             OVRDBF     FILE(STDOUT) TOFILE(QTEMP/NETCSRC@) +
                          MBR(NETCKEY)
             CALL       PGM(QNECRYPT) PARM('-e' &pr_string)
             DLTOVR     FILE(STDOUT)

/* Analyze the result and return results */
 OVRDBF     FILE(QTXTSRC) TOFILE(QTEMP/NETCSRC@) +
                          MBR(NETCKEY)

LOOP:
             RCVF
             MONMSG     MSGID(CPF0864) EXEC(GOTO CMDLBL(ENDLOOP))
             IF         COND(%SST(&srcdta 1 23) = &keyrow) THEN(DO)
                 CHGVAR     VAR(&io_enc) VALUE(%SST(&srcdta 26 25))
                 MONMSG     MSGID(MCH3601) /* Parameter not passed */
             ENDDO
             IF         COND(%SST(&SRCDTA 1 23) = &hexrow) THEN(DO)
                 CHGVAR     VAR(&io_enchex) VALUE(%SST(&srcdta 26 50))
                 MONMSG     MSGID(MCH3601) /* Parameter not passed */
             ENDDO
             GOTO       CMDLBL(LOOP)
```

Figure 445. RTVENCKEY Command CPP (Part 1 of 2)

```
                 ENDLOOP:
             DLTOVR     FILE(QTXTSRC)
             DLTF       FILE(QTEMP/NETCSRC@)
             RETURN

 STDERR:     RCVMSG     MSGTYPE(*EXCP) MSGDTA(&msgdta) +
                                       MSGID(&msgid) MSGF(&msgf) +
                                       MSGFLIB(&msgflib)
             MONMSG     MSGID(CPF0000 MCH0000)
             SNDPGMMSG  MSGID(&msgid) MSGF(&msgflib/&msgf) +
                          MSGDTA(&msgdta) MSGTYPE(*ESCAPE)
             MONMSG     MSGID(CPF0000 MCH0000)
             RETURN

             ENDPGM
```

Figure 446. RTVENCKEY Command CPP (Part 2 of 2)

A.2 Registering Overridable Functions

The following command, CL program, and QMQRY source can be used to register overridable functions to the Net.Commerce database.

```
CMD         PROMPT('Register OFS')

             PARM       KWD(INSTANCE) TYPE(*NAME) LEN(10) MIN(1) +
                          CASE(*MONO) PROMPT('Instance name')

             PARM       KWD(SRVPGM) TYPE(*NAME) LEN(10) MIN(1) +
                          PROMPT('Service program name')

             PARM       KWD(OFS) TYPE(*CHAR) LEN(32) MIN(1) +
                          CASE(*MIXED) PROMPT('Overridable function +
                          name')

             PARM       KWD(TEXT) TYPE(*CHAR) LEN(50) DFT(*BLANK) +
                          SPCVAL((*BLANK '')) CASE(*MIXED) +
                          PROMPT(Description)

             PARM       KWD(VEND) TYPE(*CHAR) LEN(32) DFT('MyCompany') +
                          CASE(*MIXED) PROMPT('Vendor Name')

             PARM       KWD(VERSION) TYPE(*DEC) LEN(3 0) DFT(1) +
                          PROMPT('OFS version')
```

Figure 447. REGOFS Command Definition

```
/*********************************************************/
/* This program will register new OFS to Net.Commerce     */
/*                                                        */
/* Arguments:                                             */
/*            pr_inst    - Instance name(library)         */
/*            pr_dll     - Dll name(Service program)      */
/*            pr_ofs     - OFS name                       */
/*            pr_text    - Description                    */
/*            pr_vend    - Vendor Name                    */
/*            pr_version - Dll version                    */
/* Author: shahar mor                                     */
/* Provided AS IS                                                */
/*********************************************************/

             PGM        PARM(&pr_inst         +
                             &pr_dll          +
                             &pr_ofs          +
                             &pr_text         +
                             &pr_vend         +
                             &pr_version      +
                                              )

/* Input parameters */
             DCL        VAR(&Pr_inst )   TYPE(*CHAR) LEN(10)
             DCL        VAR(&pr_dll  )   TYPE(*CHAR) LEN(10 )
             DCL        VAR(&pr_ofs  )   TYPE(*CHAR) LEN(32)
             DCL        VAR(&pr_text )   TYPE(*CHAR) LEN(50)
             DCL        VAR(&pr_vend )   TYPE(*CHAR) LEN(32)
             DCL        VAR(&pr_version) TYPE(*DEC ) LEN(3 0)

/* Character parameters for QMQRY REGOFS */
             DCL        VAR(&name    )   TYPE(*CHAR) LEN(34)
             DCL        VAR(&desc    )   TYPE(*CHAR) LEN(52)
             DCL        VAR(&dll     )   TYPE(*CHAR) LEN(12)
             DCL        VAR(&vend    )   TYPE(*CHAR) LEN(34)
             DCL        VAR(&vers    )   TYPE(*CHAR) LEN(3 )

             DCL        VAR(&curlib  )   TYPE(*CHAR) LEN(12)

/* General error variables */
             DCL        VAR(&msgflib)    TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgf)       TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgid)      TYPE(*CHAR) LEN(7)
             DCL        VAR(&msgdta)     TYPE(*CHAR) LEN(128)

             MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(STDERR))

/* Set the correct instance */
             RTVJOBA    CURLIB(&curlib)
             CHGCURLIB  CURLIB(&pr_inst)

/* Prepare character fields for REGOFS */
             CHGVAR     VAR(&dll) VALUE('''' *TCAT &pr_dll    *TCAT +
                          '''')

             CHGVAR     VAR(&desc) VALUE('''' *TCAT &pr_text   *TCAT +
                          '''')

             CHGVAR     VAR(&vend) VALUE('''' *TCAT &pr_vend   *TCAT +
                          '''')

             CHGVAR     VAR(&name) VALUE('''' *TCAT &pr_ofs    *TCAT +
                          '''')


             CHGVAR     VAR(&vers) VALUE(&pr_version)
```

Figure 448. REGOFS Command CPP Source (Part 1 of 2)

```
/* Register   OF to data base */
             STRQMQRY   QMQRY(REGOFS) SETVAR((DLL &dll)            +
                          (DESC &desc) (VERS &vers)                +
                             (NAME &name) (VEND &vend))

/* Restore current library */
             IF         COND(&CURLIB *NE '*NONE') THEN(DO)
                 CHGCURLIB  CURLIB(&curlib)
             ENDDO
             ELSE       CMD(DO)
                 CHGCURLIB  CURLIB(*crtdft)
             ENDDO

             RETURN
 STDERR:
             IF         COND(&curlib *NE '*NONE') THEN(DO)
                 CHGCURLIB  CURLIB(&curlib)
                 MONMSG     MSGID(CPF0000)
             ENDDO
             ELSE       CMD(DO)
                 CHGCURLIB  CURLIB(*crtdft)
                 MONMSG     MSGID(CPF0000)
             ENDDO

             RCVMSG     MSGTYPE(*EXCP) MSGDTA(&msgdta) +
                                       MSGID(&msgid) MSGF(&msgf) +
                                       MSGFLIB(&msgflib)
             MONMSG     MSGID(CPF0000 MCH0000)
             SNDPGMMSG  MSGID(&msgid) MSGF(&msgflib/&msgf) +
                          MSGDTA(&msgdta) MSGTYPE(*ESCAPE)
             MONMSG     MSGID(CPF0000 MCH0000)
             RETURN

             ENDPGM
```

Figure 449. REGOFS Command CPP Source (Part 2 of 2)

```
/* Register OFS to data base */
/* Create the QMQRY object by using command CRTQMQRY */
 insert into ofs (refnum,dll_name,vendor,product,name,version,
 description) Select max(refnum) + 1, &DLL,&VEND,'NC',&NAME,&VERS,&DESC
  from ofs
```

Figure 450. REGOFS QMQRY Source

A.3 Clear Net.Commerce Cache

The command and the CPP CLRCACH are shown in Figure 451 on page 468. CLRCACH is used to signal HTML page expiration to the cache synchronize daemon of Net.Commerce.

```
/**********************************************************/
/* The clrcache command is used to signal html page         */
/* expiration to the Net.Commerce sync daemon               */
/* Author: shahar mor                                       */
/* Provided AS IS                                           */
/**********************************************************/
             CMD        PROMPT('Clear cache')

             PARM       KWD(MERCHANT) TYPE(*DEC) LEN(9 0) MIN(1) +
                          PROMPT('Merchant number')

             PARM       KWD(PRODUCT) TYPE(*CHAR) LEN(12) DFT(*ALL) +
                          SPCVAL((*ALL ' ')) PROMPT('Product Number')
```

Figure 451. CLRCACH Command

```
H*****************************************************************************
H* This program is an example of creating file for mass import              *
     H*                                                                         *
     H* Input:                                                                  *
     H*       InMer  - Merchant number                                          *
     H*       InProd - Product number or blank for all products                 *
    H* Author: shahar mor                                                        *
   H* Provided AS IS                                               *
  H*****************************************************************************
    H                            ALWNULL(*INPUTONLY)
    F*
    F*  Products File
    FBEPROD    IF   E          K DISK    RENAME(BEPROD:RBEPROD)
    F
    F*  Products file from Net.Commerce
    FUI_PRODUCTIF   E          K DISK
    F*
    F
    F*  Cach log file
    FCACHLOG   O    E            DISK    RENAME(CACHLOG:RCACHLOG)
    D*
    D*  Parameter Structure
    D  InMer          S              9  0
    D  InProd         S             12
    D*
    D
    D*
    D*  Key for Net.Commerce product file
    D                DS
    D Kymer                          9B 0
    D KyPnbr                        64
    D*
    DAllProd          C                   CONST(' ')
  C*****************************************************************************
    C*                   Main logic
  C*****************************************************************************
    C*
    C     *ENTRY        PLIST
    C                  PARM                    InMer
    C                  PARM                    InProd
    C*
    C     KYNETC        KLIST
    C                  KFLD                    KYMER
    C                  KFLD                    KYPNBR
    C
    C*
    C                  EVAL      Kymer = InMer
    C                  EVAL      CACMENBR = InMer
    C                  EVAL      CACNAME = 'prrfnbr'
    C*
   C* Check for product existence in net commerce (Single record approach)
    C                  IF        InProd <> AllProd
    C                  EVAL      BEPNBR = InProd
    C                  EXSR      GetNbr
    C                  IF        %found
    C                  EXSR      WrtLog
    C                  ENDIF
    C                  ELSE
    C                  EXSR      WrtAll
    C                  ENDIF
    C*
    C                  EVAL      *INLR = *ON
   C*****************************************************************
    C     GetNbr        BEGSR
   C*****************************************************************
    C*
   C*  Get Net.Commerce product number from back end product number
    C*
    C                 EVAL      %subst(KyPnbr:1:12) = BEPNBR
    C     KYNETC        CHAIN(E)  UI_PRODUCT
    C*
    C                  ENDSR
```

Figure 452. CLRCACH CPP (Part 1 of 2)

```
C*********************************************************************
   C     WrtAll       BEGSR
  C*********************************************************************
   C*
   C*  Prepare all products to import file
   C*
   C     *LOVAL       SETLL     RBEPROD
   C                 READ(E)   RBEPROD
   C*
   C                 DOW       Not %EOF(BEPROD)
   C                 EXSR      GetNbr
   C                 IF        %found
   C                 EXSR      WrtLog
   C                 ENDIF
   C                 READ(E)   RBEPROD
   C                 ENDDO
   C*
   C                 ENDSR
  C*********************************************************************
   C     WrtLog       BEGSR
  C*********************************************************************
   C*
   C*  Write to cache log. Sync. daemon will do the actual purge
   C*
   C*
   C                 EVAL      CACVALUE = PRRFNBR
   C                 TIME                    CACSTMP
   C                 WRITE     RCACHLOG
   C                 ENDSR
```

Figure 453. CLRCACH CPP (Part 2 of 2)

A.4 The STRNETBE Command

The STRNETCBE command shown in Figure 454 on page 471 is used to start Net.Commerce with all the processes that are used to make back-end integration.

```
 CMD        PROMPT('Start local Net.Commerce')

             PARM       KWD(INSTANCE) TYPE(*NAME) LEN(10) MIN(1) +
                          PROMPT('Instance to start')
The STRNETCBE Command source file
/*************************************************************/
/* This program will start Net.Commerce and take care of back  */
/* end integration issues.                                     */
/*                                                             */
/* Author: shahar mor                                          */
/* Provided AS IS                                                 */
/*************************************************************/
             PGM        PARM(&pr_inst)

             DCL        VAR(&pr_inst)    TYPE(*CHAR) LEN(10)

             DCL        VAR(&libl   )    TYPE(*CHAR) LEN(275)
             DCL        VAR(&cmd    )    TYPE(*CHAR) LEN(512)

             RTVJOBA    USRLIBL(&libl)

/* Setting the back end production library to be included */

             ADDLIBLE   LIB(NETCBE ) POSITION(*LAST)
             MONMSG     MSGID(CPF2103) /* Already in libray list */
             STRNETCSVR INSTANCE(&pr_inst)

/* Starting our resdint program to bridge Net.Commerce order */
/* request to the back end system. We will add the instance  */
/* library here also. */

             ADDLIBLE   LIB(&pr_inst) POSITION(*LAST)
             MONMSG     MSGID(CPF2103) /* Already in libray list */
             SBMJOB     CMD(CALL PGM(EXTORDERR)) JOB(NETCGW)

/* Start the Net.Commerce cache daemon to make sure outdated */
/* pages are purged from the cache.                          */

             QNETCOMM/STRNETCDMN INSTANCE(&pr_inst)

/* Return the original library list */
             CHGVAR     VAR(&cmd) VALUE('chglibl libl(' *TCAT &libl +
                          *TCAT ')')
             CALL       PGM(QCMDEXC) PARM(&cmd 512)

             ENDPGM
```

Figure 454. STRNETCBE CPP Source Code

A.5 The ORDERC Program

The ORDERC program, as shown in Figure 455 on page 472, is used to actually send e-mail confirmation to the customer. It is called from the order process program.

```
    PGM        PARM(&PR_MSG &PR_ADDR)

               DCL        VAR(&PR_MSG)      TYPE(*CHAR) LEN(512)
               DCL        VAR(&PR_ADDR)     TYPE(*CHAR) LEN(246)

               SNDEMAIL   FILENAME('/shaharm/temp') +
                            RECIPADDR(&PR_ADDR) +
                            SENDERADDR(WWW@REDBOOKS.YAHOO.COM) +
                            SENDERNAME('ShopITSO orders center') +
                            SUBJECT('Your order from ShopITSO') +
                            MESSAGE(&PR_MSG)

               ENDPGM
```

Figure 455. ORDERC Source Code

A.6 Back-End Table Definition

The following SQL script describes the layout of the relevant back-end system tables:

```
CREATE TABLE NETCBE/BEMEASUR   (MSCODE CHARACTER (3) NOT NULL WITH
DEFAULT, MSTEXT CHARACTER (25) NOT NULL WITH DEFAULT, PRIMARY KEY
(MSCODE))

CREATE TABLE NETCBE/BECATEG (BECODE CHARACTER (3) NOT NULL WITH
DEFAULT, BETEXT CHARACTER (25) NOT NULL WITH DEFAULT, PRIMARY KEY
(BECODE))

CREATE TABLE NETCBE/BEPROD (BEPNBR CHARACTER (12) NOT NULL WITH DEFAULT,
BESDSC CHARACTER (25) NOT NULL WITH DEFAULT, BELDSC CHARACTER (50)
NOT NULL WITH DEFAULT, BEPRIC DEC (13, 2) NOT NULL WITH DEFAULT,
BECUR CHARACTER (3) NOT NULL WITH DEFAULT, BEGRPC CHARACTER (3) NOT
NULL WITH DEFAULT, BEINVI DEC (12) NOT NULL WITH DEFAULT, BEINVC
CHARACTER (3) NOT NULL WITH DEFAULT
       , PRIMARY KEY (BEPNBR))

LABEL ON COLUMN NETCBE/BEPROD (BEPNBR IS 'Product number', BESDSC
IS 'Short Description', BELDSC IS 'Long Description', BEPRIC IS
'Product price', BECUR IS 'Price Currency', BEGRPC IS
'Product group code', BEINVI IS 'Items in inventory', BEINVC IS
'inventory measurement code'
                              )
LABEL ON COLUMN NETCBE/BECATEG (BECODE IS 'Category code', BETEXT
IS 'Category Description')

LABEL ON COLUMN NETCBE/BEMEASUR  (MSCODE IS 'Measurement unit Code',
MSTEXT IS 'Measurements unit text')

CREATE TABLE NETCBE/BEDISC (BDPNBR CHARACTER (12) NOT NULL WITH
DEFAULT, BDCNBR DECIMAL (7) NOT NULL WITH DEFAULT, BDPCT DECIMAL
(4, 2) NOT NULL WITH DEFAULT, PRIMARY KEY (BDPNBR, BDCNBR))

LABEL ON COLUMN NETCBE/BEDISC (BDPNBR IS 'Product Number', BDCNBR
IS 'Customer Number', BDPCT IS 'Discount Percentage')

CREATE TABLE NETCBE/BEWORK (BODATE DATE NOT NULL WITH DEFAULT,
BOTIME TIME NOT NULL WITH DEFAULT, BOWS CHARACTER (10) NOT NULL
WITH DEFAULT, BOONUM CHARACTER (30) NOT NULL WITH DEFAULT, BOSUM
NUMERIC (13, 2) NOT NULL WITH DEFAULT, BOAFLG CHARACTER (1) NOT
NULL WITH DEFAULT)

 LABEL ON COLUMN NETCBE/BEWORK (BODATE IS 'Order date', BOTIME IS
 'Order Time', BOWS IS 'Order work station', BOONUM IS
 'Order Number', BOSUM IS 'Order total sum', BOAFLG IS
 'Order process flag(1=to be processed)')
```

A.7 The RQSCAP Command

The RQSCAP command is used by the back-end system to signal order fulfillment to the Net.Commerce background server.

```
/**************************************************************/

/* The RQSCAP is used by the back end system to ask the     */
/* Net.Commerce background process to capture order payment */
/* Author: shahar mor                                       */
/* Provided AS IS                                               */
/**************************************************************/
             CMD        PROMPT('Request payment capture')

             PARM       KWD(INSTANCE) TYPE(*NAME) LEN(10) MIN(1) +
                                      PROMPT('Instance name')

             PARM       KWD(MERCHANT) TYPE(*DEC) LEN(9 0) MIN(1) +
                          PROMPT('Merchant number')

             PARM       KWD(ORDER) TYPE(*DEC) LEN(9 0) MIN(1) +
                          PROMPT('Order number')

             PARM       KWD(AMOUNT) TYPE(*DEC) LEN(13 2) MIN(1) +
                          PROMPT('Order amount')
```

Figure 456. RQSCAP Command Source

```
/*********************************************************/
/* This program will request Net.Commerce background      */
/* server to issue payment capture request                */
/*                                                        */
/* Arguments:                                             */
/*            pr_inst    - Instance name                  */
/*            pr_mer     - Merchant number                */
/*            pr_ord     - Order Number                   */
/*            pr_amt     - Order amount                   */
/*                                                        */
/* Compile with  CURLIB = Net.Commerce instance library    */
/* Author: shahar mor                                     */
/* Provided AS IS                                               */
/*********************************************************/
             PGM        PARM(&pr_inst         +
                             &pr_mer          +
                             &pr_ord          +
                             &pr_amt          +
                                              )

/* Input parameters */
             DCL        VAR(&Pr_inst )   TYPE(*CHAR) LEN(10)
             DCL        VAR(&pr_mer    ) TYPE(*DEC ) LEN(9 0)
             DCL        VAR(&pr_ord    ) TYPE(*DEC ) LEN(9 0)
             DCL        VAR(&pr_amt    ) TYPE(*DEC ) LEN(13 2)

/* Character parameters for QMQRY REGOFS */

             DCL        VAR(&pr_merc)    TYPE(*CHAR) LEN(9 )
             DCL        VAR(&pr_ordc)    TYPE(*CHAR) LEN(9 )
             DCL        VAR(&pr_amtc)    TYPE(*CHAR) LEN(14)

/* General error variables */
             DCL        VAR(&msgflib)    TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgf)       TYPE(*CHAR) LEN(10)
             DCL        VAR(&msgid)      TYPE(*CHAR) LEN(7)
             DCL        VAR(&msgdta)     TYPE(*CHAR) LEN(128)

             DCL        VAR(&curlib  )   TYPE(*CHAR) LEN(12)

             MONMSG     MSGID(CPF0000) EXEC(GOTO CMDLBL(STDERR))

/* Set the correct instance */
             RTVJOBA    CURLIB(&curlib)
             CHGCURLIB  CURLIB(&pr_inst)

/* Prepare character fields for REGOFS */

             CHGVAR     VAR(&pr_merc) VALUE(&pr_mer)
             CHGVAR     VAR(&pr_ordc) VALUE(&pr_ord)
             CHGVAR     VAR(&pr_amtc) VALUE(&pr_amt)

/* Request the capture           */
             STRQMQRY   QMQRY(rqscap) SETVAR((MER &pr_merc)           +
                          (ORD &pr_ordc) (AMT &pr_amtc   ))


/* Restore current library */
             IF         COND(&CURLIB *NE '*NONE') THEN(DO)
                 CHGCURLIB  CURLIB(&curlib)
             ENDDO
             ELSE       CMD(DO)
                 CHGCURLIB  CURLIB(*crtdft)
             ENDDO
             RETURN
 STDERR:
             IF         COND(&curlib *NE '*NONE') THEN(DO)
                 CHGCURLIB  CURLIB(&curlib)
                 MONMSG     MSGID(CPF0000)
```

Figure 457. REQCAP CPP Source (Part 1 of 2)

```
             ENDDO
             ELSE       CMD(DO)
                 CHGCURLIB  CURLIB(*crtdft)
                 MONMSG     MSGID(CPF0000)
             ENDDO

             RCVMSG     MSGTYPE(*EXCP) MSGDTA(&msgdta) +
                                       MSGID(&msgid) MSGF(&msgf) +
                                       MSGFLIB(&msgflib)
             MONMSG     MSGID(CPF0000 MCH0000)
             SNDPGMMSG  MSGID(&msgid) MSGF(&msgflib/&msgf) +
                          MSGDTA(&msgdta) MSGTYPE(*ESCAPE)
             MONMSG     MSGID(CPF0000 MCH0000)
             RETURN

             ENDPGM
```

Figure 458. REQCAP CPP Source (Part 2 of 2)

```
/* Request payment capture   */
 UPDATE SETSTATUS
 SET SETSSTATCODE = 7
 WHERE setsornbr = &ORD  and
 setsmenbr = &MER        and
 setsstatcode = 21       and
 setsauthamt = &AMT
```

Figure 459. REQCAP SQL Script — QMQRY Source

A.8 HTML Samples

This section contains all of the HTML source files that we used in this book.

A.8.1 Index HTML

The Index HTML controls the frameset with two frames: the banner frame which presents the banner1.html and the main frame which presents our home page (home.html):

```
<HTML>

<HEAD>
<TITLE>Shop ITSO</TITLE>
</HEAD>

<FRAMESET ROWS="100,*" BORDER="0" FRAMEBORDER="no">
<FRAME NAME="banner" SRC="/shopitso/banner1.html" FRAMEBORDER="no" BORDERCOLOR="white"
SCROLLING="no" RESIZE="no">
<FRAME NAME="main" SRC="/shopitso/home.html" FRAMEBORDER="no">
</FRAMESET>

</HTML>
```

A.8.2 Banner1 HTML

Banner1 controls the access to our several HTML pages:

```
<html>
<head>
<meta http-equiv="Expires" content="Mon, 01 Jan 1996 01:01:01 GMT">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body bgcolor="#E00000">
```

```
<table border="0" cellpadding="0" cellspacing="0" width="640">
 <tr><td><img src="/shopitso/shopitso.gif"></td></tr>
 <tr><td><table border="1" cellspacing="0" width="640">
   <tr><td align="center" width="10%" bgcolor="#A1A2C5">
        <a href="/shopitso/home.html" target="main"><font size="1"
          face="Verdana"> <strong>Home</strong></font></a></td>
       <td align="center" width="15%" bgcolor="#E7E7EF">
        <a href="/shopitso/catalog.html" target="_top"><font size="1"
          face="Verdana"><strong>Online Shop</strong></font></a></td>
       <td align="center" width="15%" bgcolor="#A1A2C5">
         <a href="/shopitso/news.html" target="main"><font size="1"
          face="Verdana"><strong>News</strong></font></a></td>
      <td align="center" width="25%" bgcolor="#A1A2C5">
        <a href="/shopitso/company.html" target="main"><font size="1"
          face="Verdana"><strong>Our Company</strong></font></a></td>
      <td align="center" width="10%" bgcolor="#A1A2C5">
        <a href="/shopitso/help.html" target="main"><font size="1"
          face="Verdana"><strong>Help</strong></font></a></td>
      <td align="center" width="25%" bgcolor="#A1A2C5">
        <a href="/shopitso/contact.html" target="main"><font size="1"
          face="Verdana"><strong>Contact Info</strong></font></a></td>
 </tr>
</table>
</tr>
</table>
</body>
</html>
```

A.8.3 Home HTML

Our home page is shown in the main frame after the image:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<h3><font face="Helvetica">This is the <em>Home</em> Page</font></h3>
<P>Welcome to the <B>Net.Commerce ShopITSO</B>
,your sample shop for all your day-to-day needs.  <P>
For your convenience, we are open 24 hours,  7 days a week. <BR>
</body>
</html>
```

A.8.4 News HTML

Our news HTML page has a link to the IBM Net.Commerce Web site, which is on another server:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<h3><font face="Helvetica">This is the <em>News</em> Page</font></h3><BR>
<H2> Find here the latest information about Net.Commerce </H2>
<A HREF="http://www.software.ibm.com/commerce/net.commerce/" target="-1">
<img src="/shopitso/netcomm.gif" alt="Link to Net.Commerce"> Link to IBM Net.Commerce
Site</a>
</body>
</html>
```

A.8.5 Catalog HTML

The catalog HTML page starts our Net.Commerce e-business application.

It controls two framesets. The first is the banner2.html, which presents the banner and the navigation bars. The second is divided into two frames. On the left frame (named left) is our catalog tree and the frame main that presents our promotions page (promotions.html).

```
<HTML>
<HEAD>
<TITLE>Shop ITSO</TITLE>
</HEAD>
<FRAMESET ROWS="100,*" BORDER="0" FRAMEBORDER="no">
  <FRAME NAME="banner" SRC="/shopitso/banner2.html" FRAMEBORDER="no"
   BORDERCOLOR="white" SCROLLING="no" RESIZE="no">
<FRAMESET COLS="210,*" BORDER="0" FRAMEBORDER="no">
  <FRAME NAME="left" SRC="/cgi-bin/ncommerce3/CategoryDisplay?cgmenbr=28&cgrfnbr=657"
    FRAMEBORDER="no">
  <FRAME NAME="main" SRC="/shopitso/promotions.html" FRAMEBORDER="no">
</FRAMESET>
</FRAMESET>
</HTML>
```

A.8.6 Company HTML

Our company HTML page has two links: one to the Shop IBM, a real Net.Commerce application, and one to the IBM home page. Both are on other servers in the network.

```
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<h3><font face="Helvetica">This is the <em>Company</em> Page</font></h3>
<H2> Find here the latest information about our Company </H2>
<A HREF="http://www.direct.ibm.com" target="-1"><img src="/shopitso/netcomm.gif" alt="" >
Link to Shop IBM</A><BR>
<A HREF="http://www.ibm.com" target="-1"><img src="/shopitso/netcomm.gif" alt="" > Link to
IBM Homepage</A>
</body>
</html>
```

A.8.7 Help HTML

Our help HTML page has, at the moment, not much text. It will be constructed later.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<h3><font face="Helvetica">This is the <em>Help</em> Page</font></h3>
</body>
</html>
```

A.8.8 Contact HTML

For performance reasons, we did not use the contact.d2w macro, which gets the information that is shown from the database through several SQL queries. Because this information seldom changes, we use static HTML to show our shop address:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD ALIGN="left" VALIGN="center"><FONT COLOR="$(TitleTxtCol)"
   FACE="helvetica"> <H3>Contact Info</H3></FONT></TD></TR>
<TR><TD><FONT SIZE=2>For any customer needs, contact us at : </FONT><BR><BR></TD></TR>
<TR><TD><FONT SIZE=2><address>
    <B>ShopITSO</b><BR>
    <B>IBM ITSO Rochester</b><BR>
    <B>Rochester, MN</b><BR>
    <B>55555, US</b><BR><BR> <BR>
    Telephone: <B> 1-800-Call-ShopITSO>/B><BR>
    Fax: <B> 1-800-Fax_ShopITSO </B> <BR>
    Email: webmaster@shopitso.itsoroch.ibm </address></FONT></TD></TR>
</TABLE>
</body>
</html>
```

A.8.9 Banner2 HTML

The banner2 HTML controls navigation bar2. There are links to HTML pages and calls of Net.Commerce commands for OrderItemDisplay to display the order details and OrderList to show the status order page.

```
<html>
<head>
<meta http-equiv="Expires" content="Mon, 01 Jan 1996 01:01:01 GMT">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body bgcolor="#E00000">
<table border="0" cellpadding="0" cellspacing="0" width="640">
<tr><td><img src="/shopitso/shopitso.gif"></td></tr>
  <tr><td><table border="1" cellspacing="0" width="640">
  <tr><td align="center" width="10%" bgcolor="#A1A2C5">
        <a href="/shopitso/index.html" target="_top">
        <font size="1" face="Verdana"><b>Home</b></font></a></td>
       <td align="center" width="15%" bgcolor="#E7E7EF">
        <a href="/shopitso/promotions.html" target="main">
        <font size="1" face="Verdana"><b>Online Shop</b></font></a></td>
     <td align="center" width="10%" bgcolor="#E7E7EF">
       <a href="/shopitso/search.html" target="main">
       <font size="1" face="Verdana"><b>Search</b></font></a></td>
     <td align="center" width="15%" bgcolor="#E7E7EF"><a
href="/cgi-bin/ncommerce3/ExecMacro/shopitso/ordernow.d2w/input"
       target="main"><font size="1" face="Verdana">
       <b>Order Now</b></font></a></td>
     <td align="center" width="25%" bgcolor="#E7E7EF"><a
href="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=28"
       target="main"><font size="1" face="Verdana">
       <b>Display Order Details</b></font></a></td>
     <td align="center" width="25%" bgcolor="#E7E7EF"><a
href="/cgi-bin/ncommerce3/OrderList?merchant_rn=28&status=C"
       target="main"><font size="1" face="Verdana">
       <b>Check Orders Status</b></font></a></td></tr>
</table></td></tr>
</table>
</body>
</html>
```

A.8.10 Promotions HTML

The promotion page shows our special promotions for the week. It describes our special offers for the week and has a button to add a product to the order list. This is the plan for the next week. Here, we do not have any promotions:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title></title>
</head>
<body>
<h3><font face="Helvetica">This is the <em>Promotions</em> Page</font></h3>
</body>
</html>
```

A.8.11 Search HTML

In this HTML, the customer can enter a search string to search for a product name. After they press the Search button, the searchrslt.d2w presents the search result:

```
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE></TITLE>
</HEAD>
<BODY>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD ALIGN="left" VALIGN="center"><FONT SIZE=2>
Enter a keyword to quickly find the item you are looking for.<BR></FONT><BR><BR></TD></TR>
<TR><TD ALIGN="left"><FORM METHOD="POST"
ACTION="/cgi-bin/ncommerce3/ExecMacro/shopitso/searchrslt.d2w/report">
<INPUT TYPE="text" NAME="search" SIZE="30" MAXLENGTH="30"><BR> <BR>
<INPUT TYPE="submit" value="Search"></FORM></TD></TR>
</TABLE>
</BODY>
</HTML>
```

A.8.12 20BOG HTML

This HTML page is used for an additional description of the product with the SKU number 20BoG. The path and name information for this HTML page is assigned in the field PRULR of the product table. For this product, see 13.11, “Using the Product PRURL Field” on page 218.

```
<HR><TABLE width=100% cellspacing=15 border=0><TR valign=top><TD width=33%> </TD>
<TD with=67%><H1>Additional Text</h1></TD>
<TD>This is text from the URL in the database.<BR>
 We used here some HTML tags.</TD></TR>
<A HREF="/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=1450&prmenbr=28" TARGET="main">return to
product page</A>
</TABLE>
```

A.8.13 CMDINC HTML

This is the HTML file for one of the system error pages. See 13.19, “Customizing System Error Pages” on page 272.

```
<!--
  The sample templates, HTML and Macros are furnished by IBM as sample
examples to provide an illustration. These examples have not been
thoroughly tested under all conditions. IBM, therefore, cannot guarantee or
imply reliability, serviceability, of 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 names are fictitious and any similarity to the names
and addresses used by an actual business enterprise is entirely coincidental.

(C) Copyright IBM Corp. 1995, 1996
  -->
<HTML>

<HEAD>
  <TITLE>
    Mall/Store Error
  </TITLE>
</HEAD>


<BODY bgcolor="#F4F4D9">
  <br><br><br>

  <center>
    <img src="/shopitso/error.gif">
    <br><br><br>

    There is a problem in our ShopITSO store.<BR>
    Please try again later.<P>
    Perhaps you can continue to browse in our product catalog, when you see the catalog tree
in the left frame.<BR>
    Or try to use our Contact Info page, which you can reach through our Hompage (use Home
button).

    <HR WIDTH=500>
    <table border=0 width=450>
      <tr>
        <td width=450 valign=top>
          <CENTER>
            <BR><H2><B>Command Structure Failure</B></H2><P>
            (CMN0950E)<P>
            Unable to complete command. The command syntax is
            not correct, or parameters required by the system
            were not passed in.
          </CENTER>
        </td>
    </table>

  </center>

</BODY>

</HTML>
```

A.9 Net.Data Sample Macros

This section contains all of the Net.Data macro source files that we used in this book.

A.9.1 Macro for Catalog Tree

This is the macro to build our catalog tree, which is shown when the online shop is started form the navigation bar 1 in the left frame. The name of the macro is cat0.d2w and is assigned to the Top Category (Home Category) of our store.

```
%include "ShopITSO/ShopITSO.inc"

%{=====================================================================

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-D245798-NC3

(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

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

%define {
  SHOWSQL="NO"
SHIPPING_REF = "0"
rootTable = %table
   rowIndexRoot = "1"
   parentH1=""
   catChild=""
%}

%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
    %}
%}

%{== all categories under top category ==%}
%function(dtw_SQL) GET_CATEGORY_ROOT1(OUT table) {
   SELECT CGRFNBR, CGNAME, CGFIELD1
   from category, cgryrel
   where  CATEGORY.CGRFNBR = CGRYREL.CRCCGNBR and CATEGORY.
          CGMENBR = $(MerchantRefNum) and CGRYREL.CRPCGNBR = $(HomeCategory)
          and cgpub=1
   %REPORT {
     %ROW {
     %}
   %}
   %MESSAGE {
          default: { %} :continue  %}
%}

%{== all categories under root1 category ==%}
%function(dtw_SQL) GET_CATEGORY_CHILD() {
   SELECT CGRFNBR, CGNAME
   from category, cgryrel
   where  CATEGORY.CGRFNBR = CGRYREL.CRCCGNBR and
          CATEGORY.CGMENBR = $(MerchantRefNum) and
          CGRYREL.CRPCGNBR = $(parentH1) and cgpub=1
   %REPORT {
    <UL>
     %ROW {
       @DTW_ASSIGN(catChild, V_cgrfnbr)
      <LI><A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgmenbr=$(MerchantRefNum)&cgrfnbr=$(V_cgrfnb
r)"
       TARGET="left"><B><FONT SIZE=2>$(V_cgname)</FONT></B></A>
     %}
   </UL>
%}
%MESSAGE {
          default: { %} :continue  %}
%}

%{== macro for loop ==%}
%macro_function GET_ROOT_INDEX (IN table){
%while (rowIndexRoot <=  numRows) {
 @DTW_ASSIGN(parentH1, @DTW_TB_RGETV(rootTable, rowIndexRoot, "1"))
 <B><FONT SIZE=3>@DTW_TB_RGETV(rootTable, rowIndexRoot, "2")</FONT></B>
 %IF (@DTW_TB_RGETV(rootTable, rowIndexRoot, "3") == "1")
 <A
HREF="/servlet/icviewer/ca_html/shopitso_pe.html?cgrfnbr=$(parentH1)&cgmenbr=$(Merchan
tRefNum)" TARGET="main">
  <IMG SRC="/shopitso/padvisor.gif" ALT="Product Exploration" BORDER="1"
    ALIGN="bottom"></A>
  %ENDIF
@GET_CATEGORY_CHILD()
@DTW_ADD(rowIndexRoot, "1", rowIndexRoot)
%}
%}

%function(dtw_odbc) DISPLAY_BACKUP(){
    select distinct crpcgnbr, cgname
    from cgryrel, category
    where crccgnbr=$(cgrfnbr) and crmenbr=$(MerchantRefNum)
    and cgrfnbr=crpcgnbr

  %REPORT{

  <TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
    %ROW{
     <TR><TD ALIGN="left" VALIGN="top">
    %IF (V_cgrfnbr != $(HomeCategory))
    <A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_crpcgnbr)&cgmenbr=$(MerchantRefN
um)">
   <B><FONT COLOR=$(TitleTxtCol) SIZE=2>RETURN TO $(V_cgname)</FONT></B></A>
   %ENDIF
```

```
</TD></TR>
    %}
</TABLE>

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

%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY BGCOLOR="#E00000" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
@GET_ADDRESS_REF_NUM()
<TABLE BGCOLOR="$(BodyColor1)" BORDER=0 CELLPADDING=3 CELLSPACING=0 WIDTH=100%>
<TR><TD ALIGN="left"><FONT FACE="helvetica"
COLOR="$(TitleTxtCol)"><H5>Catalog</H5></FONT></TD></TR>
<TR><TD ALIGN="left"><FONT COLOR="$(TitleTxtCol)" SIZE=2>
Browse the catalog to view product information and to add items to the order list, or use
Product Advisor Tools (indicated by a
<IMG SRC="/shopitso/padvisor.gif" ALIGN="middle"> beside a product category) to select
products by its distinctive attributes.</A>
</FONT><BR></TD></TR>
<TR><TD ALIGN="left">
@DISPLAY_BACKUP()</TD></TR>
</TABLE>
@GET_CATEGORY_ROOT1(rootTable)
@DTW_TB_ROWS(rootTable, numRows)
<TABLE BGCOLOR="$(BodyColor1)" WIDTH=100% CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR><TD>@GET_ROOT_INDEX(rootTable)</TD></TR>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.2 Category Macro

All of our categories, except the Home Category, have the macro cat1.2dw assigned:

```
%include "ShopITSO/ShopITSO.inc"
%{======================================================================
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-D245798-NC3

(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
=====================================================================%}
%define {
  SHOWSQL="NO"
  SHIPPING_REF = "0"
  AATTRIBUTES = "FALSE"
  BACKUP = "$(HomeCategory)"
  CGRYNUM = ""
  CGRYBANNERNAME = ""
%}

%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) DISPLAY_BACKUP(){
    select distinct crpcgnbr, cgname
    from cgryrel, category
    where crccgnbr=$(cgrfnbr) and crmenbr=$(MerchantRefNum)
          and cgrfnbr=crpcgnbr
%REPORT{
   %ROW{
<A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_crpcgnbr)&cgmenbr=$(MerchantRefN
um)"><B>
    <FONT COLOR=$(TitleTxtCol) SIZE=2>RETURN TO $(V_cgname)</FONT></B></A>
%}
   %}
  %MESSAGE{100:{%} :continue %}
%}

%{==== DISPLAY_CATEGORIES Function ====%}
%function(dtw_odbc) DISPLAY_CATEGORIES(){
    select CATEGORY.CGRFNBR, CATEGORY.CGMENBR, CATEGORY.CGNAME,
           CATEGORY.CGTHMB, CATEGORY.CGFIELD1, 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{
   <B><UL>
   %ROW{
     <LI><A
HREF="/cgi-bin/ncommerce3/CategoryDisplay?cgrfnbr=$(V_CGRFNBR)&cgmenbr=$(V_CGMENBR)&CG
RY_NUM=$(CGRYNUM)">
     <FONT SIZE=2>$(V_CGNAME)</FONT></A>
     %IF (V_CGFIELD1 == "1")
<A
HREF="/servlet/icviewer/ca_html/shopitso_pe.html?cgrfnbr=$(V_CGRFNBR)&cgmenbr=$(Mercha
ntRefNum)" TARGET="main">
      <IMG SRC="/shopitso/padvisor.gif" ALT="Product Exploration"
           BORDER="1" ALIGN="bottom"></A>
     %ENDIF
</LI>
@DTW_ASSIGN(save_crpcgnbr, V_CRPCGNBR)
   %}
@DTW_assign(BACKUP, save_crpcgnbr)
</UL></B>
  %}
  %MESSAGE{100:{%} :continue %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_LIST() {
    select PRODUCT.PRRFNBR, PRODUCT.PRMENBR, PRODUCT.PRTHMB,
           PRODUCT.PRNBR, PRSDESC, PRTHMB,
           CGPRREL.CPSEQNBR, CGPRREL.CPCGNBR
    from PRODUCT, CGPRREL
    where CPPRNBR=PRRFNBR and cpcgnbr=$(cgrfnbr) and cpmenbr=$(cgmenbr)
          and prpub=1
    order by cpseqnbr
  %REPORT{
     <B><UL>
    %ROW{
     <LI><A
HREF="/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=$(V_PRRFNBR)&prmenbr=$(V_PRMENBR)"
TARGET="main">
     <FONT SIZE=2><B><I>$(V_PRSDESC)</I></B></FONT></A></LI>
@DTW_ASSIGN(save_crpcgnbr, V_CRPCGNBR)
%}
@DTW_assign(BACKUP, save_crpcgnbr)
</UL></B>
  %}
  %MESSAGE{100:{%} :continue %}
%}

%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY BGCOLOR="#E00000" TEXT="$(TextCol)" LINK="$(LinkCol)"
  VLINK="$(VLinkCol)" ALINK="$(ALinkCol)">
<TABLE BGCOLOR="$(BodyColor1)" BORDER=0 CELLPADDING=3 CELLSPACING=0
  WIDTH=100%>
<TR><TD ALIGN="left"><FONT FACE="helvetica"
COLOR="$(TitleTxtCol)"><H3>Catalog</H3></FONT></TD></TR>
<TR><TD ALIGN="left">
<FONT COLOR="$(TitleTxtCol)" SIZE=2>
Browse the catalog to view product information and to add items to the order list, or use
Product Advisor Tools (indicated by a <IMG
SRC="/shopitso/padvisor.gif" ALIGN="middle"> beside a product category) to select products by
its distinctive attributes.</A>
</FONT><BR><BR></TD></TR>
%IF (V_cgrfnbr != $(HomeCategory))
<TR><TD ALIGN="left">
@DISPLAY_BACKUP()</TD></TR>
%ENDIF
</TABLE>
<BR>
@GET_ADDRESS_REF_NUM()
<TABLE BGCOLOR="$(BodyColor1)" WIDTH=100% CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR><TD>@DISPLAY_CATEGORIES()</TD></TR>
<TR><TD>@DISPLAY_PRODUCT_LIST()</TD></TR>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.3 Product Macro PROD1.D2W

This section contains the product macro named prod1.d2w. This template is assigned to all of our products, with the exception of the product with the SKU number 20BOG.

```
%include "ShopITSO/ShopITSO.inc"
%{====================================================================
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-D245798-NC3

(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
====================================================================%}
%define {
  SHOWSQL="YES"
  SHIPPING_REF="0"
   ATTRIBUTES = "FALSE"
  ITEM_ATTR_NAME = ""
  ADDRESS_REF = ""
  DESC1=""
  DESC2=""
  DESC3=""
%}

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

   %REPORT{
     %ROW{
      %IF (ROW_NUM == "1" && SHIPPING_REF == "0")
  @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ELIF (ROW_NUM == "2" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ELIF (ROW_NUM == "3" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ENDIF
     %}
   %}
   %MESSAGE{
     default: {SHIPPING MODE ERROR %}: continue
   %}
%}

%function(dtw_odbc) CHECK_PRODUCT_ATTR() {
   selectpdname
   from proddstatr
   where pdprnbr=$(prrfnbr) and pdmenbr=$(MerchantRefNum)
   %REPORT{
     %ROW{
     @DTW_assign(ATTRIBUTES, "TRUE")
     %}
   %}
   %MESSAGE{
     default: { %}: continue
   %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_IMAGE(){
    SELECT prthmb, prfull, prsdesc, prldesc1, prldesc2, prldesc3, prnbr
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr)
  %REPORT{
    %ROW{
       <TR><TD ALIGN="center" BGCOLOR="$(BodyColor2)">
           <FONT SIZE=2 COLOR="$(TitleTxtCol)">
           <B>$(V_PRSDESC)</TD><BR><BR>
     <TD ALIGN="left" BGCOLOR="$(BodyColor2)">
       <B>SKU: $(V_prnbr)</B></FONT></TD></TR> <BR> <BR> <BR>
     %IF (V_prfull != "")
     <TR><TD COLSPAN=2 ALIGN="left"><IMG SRC="$(V_prfull)"></TD></TR><BR>
     %ELIF (V_prthmb != "")
     <TR><TD COLSPAN=2 ALIGN="left"><IMG SRC="$(V_prthmb)"></TD></TR><BR>
        %ELSE
     <TR><TD COLSPAN=2 ALIGN="left"><B><I>Sorry, An image of the product is not
available.</I></B></TD></TR><BR>
        %ENDIF
     @DTW_assign(DESC1, V_prldesc1)
     @DTW_assign(DESC2, V_prldesc2)
```

```
      @DTW_assign(DESC3, V_prldesc3)
      %}
       %IF (ATTRIBUTES == "FALSE")
       <TR>
       <FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="main" METHOD="post">
        <TD> Please type quantity you want to order:
        <INPUT TYPE=text NAME=quantity VALUE=1 SIZE=5 MAXLENGTH=32> </TD>
    <INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
    <INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
    <INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
    <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
    <TD ALIGN="left" COLSPAN=2><input type=image SRC="$(AddButton)"></TD></TR>
      </FORM>
<TR><TD><BR></TD></TR>
%ENDIF
</TABLE>
  %}
  %MESSAGE{100:{ %} :continue %}
%}



%function(dtw_odbc) DISPLAY_PRODATTR_VALUES(){
    SELECT distinct paname, paval
    FROM PRODUCT, PRODATR, PRODDSTATR
    WHERE pamenbr=$(MerchantRefNum) and prmenbr=$(MerchantRefNum)
          and paprnbr=prrfnbr and  prprfnbr=$(prrfnbr) and paname=pdname
    %REPORT{
     <TR>
     <FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="main" METHOD="post">
       <TD> Please type quantity you want to order:
       <TD> <INPUT TYPE=text NAME=quantity VALUE=1 SIZE=5 MAXLENGTH=32> </TD>
    <INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
    <INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
    <INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
    <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
       <BR></TD></TR>
     %ROW{
  %IF (ITEM_ATTR_NAME != V_paname)
</SELECT>
@DTW_assign(ITEM_ATTR_NAME, V_paname)
<TR><TD ALIGN="right"><B>$(V_paname)</B> </TD>
 <TD>
 <SELECT NAME="$(V_paname)"><OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>
 %ELSE
 <OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>
 %ENDIF
    %}
</SELECT>
</TD></TR>
<TR><TD><BR></TD></TR>

<TR><TD ALIGN="center" COLSPAN=2><input type=image SRC="$(AddButton)"></TD></TR>
</FORM>
<TR><TD><BR></TD></TR>
  %}
  %MESSAGE{100:{ PROBLEM%} :continue %}
%}

%{=================================================%}
%{ HTML Report Section
%{=================================================%}
%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=2 WIDTH=100%>
@GET_ADDRESS_REF_NUM()
@GET_SHIPPING_REF_NUM()
@CHECK_PRODUCT_ATTR()
@DISPLAY_PRODUCT_IMAGE()
%IF (ATTRIBUTES == "TRUE")
@DISPLAY_PRODATTR_VALUES()
%ENDIF
```

```
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR BGCOLOR="$(BodyColor2)"><TD ALIGN="center"><FONT SIZE=2><B>Price : </B> $(price)
$(currency)</FONT>
</TD> </TR><BR><P>
<TR> <TD COLSPAN=2> <FONT SIZE=2> $(DESC1)<BR> $(DESC2)<BR> $(DESC3)<BR>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.4 Product Macro PROD2.D2W

This section contains the product macro named prod2.d2w for the product with the SKU number 20BOG. This template is assigned only for this product.

It uses the PRURL field to link to a second page, the 20BOG.HTML. On this page, the additional product description is shown. See A.8.12, “20BOG HTML” on page 479.

```
%include "ShopITSO/ShopITSO.inc"
%{=====================================================================
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-D245798-NC3

(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
=====================================================================%}
%define {
  SHOWSQL="YES"
SHIPPING_REF="0"
ATTRIBUTES = "FALSE"
ITEM_ATTR_NAME = ""
ADDRESS_REF = ""
      DESC1=""
      DESC2=""
      DESC3=""
      URL=""
%}

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

   %REPORT{
     %ROW{
      %IF (ROW_NUM == "1" && SHIPPING_REF == "0")
  @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ELIF (ROW_NUM == "2" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ELIF (ROW_NUM == "3" && SHIPPING_REF == "0")
   @DTW_assign(SHIPPING_REF, V_spmmnbr)
%ENDIF
     %}
   %}
   %MESSAGE{
     default: {SHIPPING MODE ERROR %}: continue
   %}
%}

%function(dtw_odbc) CHECK_PRODUCT_ATTR() {
   selectpdname
   from proddstatr
   where pdprnbr=$(prrfnbr) and pdmenbr=$(MerchantRefNum)
   %REPORT{
     %ROW{
     @DTW_assign(ATTRIBUTES, "TRUE")
     %}
   %}
   %MESSAGE{
     default: { %}: continue
   %}
%}


%function(dtw_odbc) DISPLAY_PRODUCT_IMAGE(){
    SELECT prthmb, prfull, prsdesc, prldesc1, prldesc2, prldesc3, prnbr, prurl
    FROM product
    WHERE prmenbr=$(MerchantRefNum) and prrfnbr=$(prrfnbr)
  %REPORT{
    %ROW{
       <TR><TD ALIGN="center" BGCOLOR="$(BodyColor2)"><FONT SIZE=2 COLOR="$(TitleTxtCol)">
          <B>$(V_PRSDESC)</TD><BR><BR>
     <TD ALIGN="left" BGCOLOR="$(BodyColor2)">
       <B>SKU: $(V_prnbr)</B></FONT></TD></TR> <BR> <BR> <BR>
     %IF (V_prfull != "")
     <TR><TD COLSPAN=2 ALIGN="left"><IMG SRC="$(V_prfull)"></TD></TR><BR>
     %ELIF (V_prthmb != "")
     <TR><TD COLSPAN=2 ALIGN="left"><IMG SRC="$(V_prthmb)"></TD></TR><BR>
        %ELSE
     <TR><TD COLSPAN=2 ALIGN="left"><B><I>Sorry, An image of the product is not
available.</I></B></TD></TR><BR>
        %ENDIF
     @DTW_assign(DESC1, V_prldesc1)
     @DTW_assign(DESC2, V_prldesc2)
     @DTW_assign(DESC3, V_prldesc3)
     @DTW_assign(URL, V_prurl)
     %}
%IF (ATTRIBUTES == "FALSE")
   <TR>
   <FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="main" METHOD="post">
       <TD> Please type quantity you want to order:
       <INPUT TYPE=text NAME=quantity VALUE=1 SIZE=5 MAXLENGTH=32> </TD>
    <INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
    <INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
    <INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
    <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
    <TD ALIGN="left" COLSPAN=2><input type=image SRC="$(AddButton)"></TD></TR>
      </FORM>
<TR><TD><BR></TD></TR>
%ENDIF
</TABLE>
  %}
  %MESSAGE{100:{ %} :continue %}
%}



%function(dtw_odbc) DISPLAY_PRODATTR_VALUES(){
    SELECT distinct paname, paval
    FROM PRODUCT, PRODATR, PRODDSTATR
    WHERE pamenbr=$(MerchantRefNum) and prmenbr=$(MerchantRefNum)
          and paprnbr=prrfnbr and  prprfnbr=$(prrfnbr)and paname=pdname
  %REPORT{
      <TR>
<FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" TARGET="main"  METHOD="post">
       <TD> Please type quantity you want to order:
       <TD> <INPUT TYPE=text NAME=quantity VALUE=1 SIZE=5 MAXLENGTH=32> </TD>
    <INPUT TYPE=hidden NAME=merchant_rn VALUE=$(MerchantRefNum)>
    <INPUT TYPE=hidden NAME=product_rn VALUE=$(prrfnbr)>
    <INPUT TYPE=hidden NAME=shipmode_rn VALUE=$(SHIPPING_REF)>
    <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
       <BR></TD></TR>
     %ROW{
  %IF (ITEM_ATTR_NAME != V_paname)
</SELECT>
@DTW_assign(ITEM_ATTR_NAME, V_paname)
<TR><TD ALIGN="right"><B>$(V_paname)</B> </TD>
 <TD>
 <SELECT NAME="$(V_paname)"><OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>
 %ELSE
 <OPTION VALUE="$(V_paval)">$(V_paval)</OPTION>
 %ENDIF
    %}
</SELECT>
</TD></TR>
<TR><TD><BR></TD></TR>

<TR><TD ALIGN="center" COLSPAN=2><input type=image SRC="$(AddButton)"></TD></TR>
</FORM>
<TR><TD><BR></TD></TR>
  %}
  %MESSAGE{100:{ PROBLEM%} :continue %}
%}

%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=2 WIDTH=100%>
@GET_ADDRESS_REF_NUM()
@GET_SHIPPING_REF_NUM()
@CHECK_PRODUCT_ATTR()
@DISPLAY_PRODUCT_IMAGE()
%IF (ATTRIBUTES == "TRUE")
@DISPLAY_PRODATTR_VALUES()
%ENDIF
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR BGCOLOR="$(BodyColor2)"><TD ALIGN="center"><FONT SIZE=2><B>Price : </B> $(price)
$(currency)</FONT>
</TD> </TR><BR><P>
<TR> <TD COLSPAN=2> <FONT SIZE=2> $(DESC1)<BR> $(DESC2)<BR> $(DESC3)<BR>
<TR><TD COLSPAN=2 ALIGN="left"><A HREF="$(URL)">Get more description of this
product</A></TD></TR><BR>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.5 Macro for Current Order

This macro with the name shipto.d2w is used to show the current order and to update or delete an item in the list. The order can also be placed from this page.

```
%include "ShopITSO/ShopITSO.inc"
%{=====================================================================
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-D245798-NC3

(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
=========================================================================%}
%define {
   SHOWSQL="NO"
   SHOPPER_REF=""
   fgrandtot=""
   GRAND_TOT = "0"
   SUB_TOT = ""
%}

%{==== Retrieves the Shopper Reference Number ====%}
%function(dtw_odbc) GET_SHOPPER_REF_NUM() {
   select shrfnbr from shopper where shlogid = '$(SESSION_ID)'
   %REPORT{
     %ROW{
       @DTW_assign(SHOPPER_REF, V_shrfnbr)
     %}
   %}
   %MESSAGE{
     default: { ERROR in GET_SHOPPER_REF_NUM %}
   %}
%}



%function(dtw_odbc) GET_TOTAL_DETAILS() {
SELECTstprice, stquant, stcpcur, strfnbr, stsanbr, ststat,
      stsmnbr, prsdesc, prrfnbr, prnbr
FROMshipto, product
WHEREstmenbr=$(MerchantRefNum) and stshnbr=$(SHOPPER_REF) and stprnbr=prrfnbr and ststat='P'
   %REPORT{
  <TABLE WIDTH=530 BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TR><TD ALIGN="left"  COLSPAN=3><FONT SIZE=2>
The following table displays the
items that you added to the Order List.<BR> <BR>
For each item, you can change the quantity, or you can remove the item.<BR>
Please make your changes for every row separat.</FONT><BR><BR></TD></TR>
</TABLE>
<TABLE WIDTH=530 BORDER=1 CELLPADDING=0 CELLSPACING=0>
<TR><TD><b><FONT SIZE=2> Item </FONT> </b></TD>
 <TD><b><FONT SIZE=2> Quantity </FONT> </b></TD>
 <TD><b><FONT SIZE=2> Cost per Item </FONT> </b></TD>
 <TD><b><FONT SIZE=2> Subtotal</FONT> </b></TD>
</TR>
%ROW{
@DTW_MULTIPLY(V_stquant, V_stprice, SUB_TOT)
@DTW_ADD(SUB_TOT, GRAND_TOT, GRAND_TOT)
<TR ALIGN="left">
<FORM ACTION="/cgi-bin/ncommerce3/OrderItemUpdate" METHOD="POST" TARGET="main">
<INPUT TYPE="hidden" NAME="merchant_rn" value="$(MerchantRefNum)">
<INPUT TYPE="hidden" NAME=shipto_rnVALUE="$(V_strfnbr)">
<INPUT TYPE="hidden" NAME=shipmode_rnVALUE="$(V_stsmnbr)">
<INPUT TYPE="hidden" NAME="url"
VALUE="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
</TD>
<TD><FONT SIZE=2>$(V_prsdesc)</FONT></TD>
<TD><FONT SIZE=3><INPUT TYPE="field" NAME="quantity"  VALUE="$(V_stquant)"
SIZE=2></FONT></TD>
<TD ALIGN="right"><FONT SIZE=2> $(V_stprice) $(V_stcpcur) </FONT></TD>
<TD ALIGN="right"><FONT SIZE=2> $(SUB_TOT) $(V_stcpcur) </FONT></TD>
<TD>
<INPUT TYPE="SUBMIT" VALUE="Update">
</FORM>
</TD>
<TD>
<FORM ACTION="/cgi-bin/ncommerce3/OrderItemDelete" METHOD="POST" TARGET="main">
<INPUT TYPE="hidden" NAME="shipto_rn" VALUE="$(V_strfnbr)">
 <INPUT TYPE=hidden name="url"
value="/cgi-bin/ncommerce3/OrderItemDisplay?merchant_rn=$(MerchantRefNum)">
```

```
         <INPUT TYPE="submit" VALUE="Remove">
</FORM>
</TD>
</TR>
    %}
</TABLE>
<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>
 <TR><TD ALIGN="left" VALIGN="center">
   <FORM ACTION="/cgi-bin/ncommerce3/OrderDisplay" TARGET="main">
<INPUT TYPE=hidden name="status" value="P">
<INPUT TYPE=hidden name="merchant_rn" value="$(MerchantRefNum)">
<TD ALIGN="center" WIDTH="100"><INPUT TYPE="submit" VALUE="Place Order"></TD>
</FORM></TD></TR>
</TABLE> <BR> <BR>
<TABLE WIDTH=530 BORDER=2 CELLPADDING=0 CELLSPACING=0>
 <TR ALIGN="center" BGCOLOR="white"><TD COLSPAN=3><FONT SIZE=2><SPACER TYPE="vertical"
SIZE=5>
<B>Total</b> : $(GRAND_TOT) $(CURRENCY)<I>(Before Taxes and Shipping)</I>
<SPACER TYPE="vertical" SIZE=5></FONT></TD></TR>
</TABLE>
%}
 %MESSAGE{
     default: {
<FONT SIZE=2>All items have been removed. Continue shopping to add more items to your
Order List.</FONT>
%}:continue
 %}
%}
%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT{
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
@GET_SHOPPER_REF_NUM()
<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>
 <TR><TD ALIGN="left" VALIGN="center"><FONT FACE="helvetica" COLOR=$(TitleTxtCol)><H3>Order
Details</H3></FONT>
 </TD></TR>
</TABLE>
@GET_TOTAL_DETAILS()
<BR><BR>
</BODY>
</HTML>
%}
```

A.9.6 Macro for Accepted the Order (Alternative 1)

This macro named order.d2w is shown when the customer submits the order. Here, we also work with SET protocol.

```
%include "ShopITSO/ShopITSO.inc"
%{=====================================================================

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-D245798-NC3

(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
```

```
=========================================================================%}
%define {
 SHOWSQL="NO"
 MERCHANT_TAX="TRUE"
 SHIPPING_NUM = "0"
 SHIPPER1_RNBR = ""
 SHIPPER2_RNBR = ""
 SHIPPER3_RNBR = ""
 SHIPPER1 = ""
 SHIPPER2 = ""
 SHIPPER3 = ""

%}

%function(dtw_odbc) GET_SHIPPING_ADDRESS_INFO() {
select sarfnbr, safname, salname, saaddr1, saaddr2, sacity, sazipc, sacntry, sastate,
saphone1, saemail1
from shaddr, shipto
where sanick='$(SESSION_ID)' and saadrflg='P' and STSANBR=sarfnbr
%report {
 %ROW{
       @DTW_assign(SHOPPER_REF, V_sarfnbr)
 @DTW_assign(save_sarfnbr, V_sarfnbr)
 @DTW_assign(save_safname, V_safname)
 @DTW_assign(save_salname, V_salname)
 @DTW_assign(save_saaddr1, V_saaddr1)
 @DTW_assign(save_saaddr2, V_saaddr2)
 @DTW_assign(save_sacity, V_sacity)
 @DTW_assign(save_sastate, V_sastate)
 @DTW_assign(save_sazipc, V_sazipc)
 @DTW_assign(save_sacntry, V_sacntry)
 @DTW_assign(save_saphone1, V_saphone1)
 @DTW_assign(save_safax, V_safax)
 @DTW_assign(save_email1, V_email1)
 %}
<INPUT TYPE=hidden NAME="sarfnbr" VALUE="$(save_sarfnbr)">
   <TABLE WIDTH=400 CELLPADDING=0  CELLSPACING=0 BORDER=0>
   <TR>
  <TD ALIGN="left"><FONT SIZE=2>First Name</FONT></TD>
<TD ALIGN="left"><FONT SIZE=2><b>Last Name</b></FONT></TD>
</TR>
   <TR>
<TD COLSPAN=1><INPUT TYPE="text" NAME="safname" VALUE="$(save_safname)" VALUESIZE="28"
MAXLENGTH="30"></TD>
<TD COLSPAN=1><INPUT TYPE="text" NAME="salname" VALUE="$(save_salname)" SIZE="28"
MAXLENGTH="30"></TD>
</TR>
   <TR> <TD ALIGN="left"><FONT SIZE=2><b>Address</b></FONT></TD> </TR>
<TR> <TD COLSPAN=3><INPUT TYPE="text" NAME="saaddr1" VALUE="$(save_saaddr1)" SIZE="62"
MAXLENGTH="50"></TD> </TR>
<TR> <TD COLSPAN=3><INPUT TYPE="text" NAME="saaddr2" VALUE="$(save_saaddr2)" SIZE="62"
MAXLENGTH="50"></TD> </TR>
   <TR> <TD ALIGN=left><FONT SIZE=2><b>City</FONT></b></TD>
  <TD ALIGN=left><FONT SIZE=2><b>State/Province</FONT></b></TD>
</TR>
   <TR> <TD COLSPAN=1><INPUT TYPE="text" NAME="sacity" VALUE="$(save_sacity)" SIZE="28"
MAXLENGTH="30"></TD>
  <TD COLSPAN=1><INPUT TYPE="text" NAME="sastate" VALUE="$(save_sastate)" SIZE="28"
MAXLENGTH="20"></TD>
</TR>
   <TR> <TD ALIGN=left><FONT SIZE=2><b>ZIP/Postal code</b></FONT></TD>
  <TD ALIGN=left><FONT SIZE=2><b>Country</b></FONT></TD>
</TR>
   <TR> <TD COLSPAN=1><INPUT TYPE="text" NAME="sazipc" VALUE="$(save_sazipc)" SIZE="28"
MAXLENGTH="20"></TD>
  <TD COLSPAN=1><INPUT TYPE="text" NAME="sacntry" VALUE="$(save_sacntry)" SIZE="28"
MAXLENGTH="30"></TD>
</TR>
   <TR> <TD ALIGN=left><FONT SIZE=2>Phone Number</FONT></TD>
  <TD ALIGN=left><FONT SIZE=2>E-mail Address</FONT></TD>
</TR>
   <TR> <TD COLSPAN=1><INPUT TYPE="text" NAME="saphone1" VALUE="$(save_saphone1)" SIZE="28"
MAXLENGTH="30"></TD>
  <TD COLSPAN=3><INPUT TYPE="text" NAME="saemail1" VALUE="$(save_saemail1)" SIZE="28"
MAXLENGTH="254"></TD>
</TR>
   </TABLE>
```

```
%}
%MESSAGE{
   100: { No Address Found!<p>
   %}
   default: {
<font size=+3><b>Database Error:</b></font><BR>
A database error occurred. Please contact the merchant server administrator.<BR>
SQL error code = $(SQL_CODE)
   %}
%}
%}

%function(dtw_odbc) DETERMINE_SHIPPING_LIST() {
select SPRFNBR, SPCHRGE, SMRFNBR , SMCARRID
fromSHIPPING, SHIPMODE
whereSPMENBR=$(MerchantRefNum) and SPRFNBR =SMRFNBR
order by spchrge
%REPORT{
%ROW{
  @DTW_ASSIGN(SHIPPING_NUM, ROW_NUM)
%IF (ROW_NUM == "1")
@DTW_ASSIGN(SHCOST1, V_SPCHRGE)
@DTW_CONCAT(" -- ", V_SPCHRGE, COST1)
@DTW_CONCAT(V_SMCARRID, COST1, SHIPPER1)
@DTW_ASSIGN(SHIPPER1_RNBR, V_SMRFNBR)
%ENDIF
%IF (ROW_NUM == "2")
@DTW_ASSIGN(SHCOST2, V_SPCHRGE)
@DTW_CONCAT(" -- ", V_SPCHRGE, COST2)
@DTW_CONCAT(V_SMCARRID, COST2, SHIPPER2)
@DTW_ASSIGN(SHIPPER2_RNBR, V_SMRFNBR)
%ENDIF
%IF (SHIPPING_NUM == "3")
@DTW_ASSIGN(SHCOST3, V_SPCHRGE)
@DTW_CONCAT(" -- ", V_SPCHRGE, COST3)
@DTW_CONCAT(V_SMCARRID, COST3, SHIPPER3)
@DTW_ASSIGN(SHIPPER3_RNBR, V_SMRFNBR)
%ENDIF
%}
%}
%MESSAGE{
default: {ERROR : Problem with DISPLAY_SHIPPING_LIST function %}
%}
%}


%function(dtw_odbc) SHOW_SUBTOTAL_PRICE_MerchantTax() {
select distinct sanick, salname, safname, oyprtot, oyshtot, oycpcur,
mttaxrate1, mttaxname1, mttaxrate2, mttaxname2, mttaxrate3, mttaxname3,
mttaxrate4, mttaxname4, mttaxrate5, mttaxname5, mttaxrate6, mttaxname6, mtmenbr,
oytax1,oytax2, oytax3, oytax4, oytax5, oytax6,
(mttaxrate1+mttaxrate2+mttaxrate3+mttaxrate4+mttaxrate5+mttaxrate6) as mttax,
         (oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as oytax, oysanbr,
         (oyprtot+oyshtot+oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as subtot1
from orderpay, shaddr, shopper, merchanttax
where shopper.shlogid='$(SESSION_ID)' and orderpay.oyornbr=$(order_rn) and
         shaddr.sashnbr=shopper.shrfnbr and shaddr.sarfnbr=orderpay.oysanbr
   and mtmenbr=$(MerchantRefNum)
order by sanick

%REPORT{
   <FORM ACTION="/cgi-bin/ncommerce3/OrderShippingUpdate">
      <INPUT TYPE=hidden NAME=order_rn VALUE= $(order_rn)>
      <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderDisplay?status=P&merchant_rn=$(MerchantRefNum)">
      <TABLE  BORDER=0 CELLSPACING=1 CELLPADDING=1 WIDTH=400 ALIGN="left" COLS=3 >
      <TR>
<TD width=200> </TD>
<TD width=180>  </TD>
<TD width=120> </TD>
  </TR>
%ROW {
@DTW_format(V_oyshtot, "", "2", V_foyshtot)
@DTW_format(V_oytax,  "", "2", V_foytax)
@DTW_format(V_mttax,  "", "2", V_fmttax)
      <TR> <TD ALIGN=right > <B>Sub Total</B> </TD>
  <TD ALIGN=right > <B> $(V_oyprtot) $(V_oycpcur)</B></TD>
   </TR>
%if ($(V_mttaxname1) != "")
      <TR> <TD ALIGN=right > $(V_mttaxname1) ($(V_mttaxrate1)%) </TD>
  <TD ALIGN=right > $(V_oytax1) $(V_oycpcur)</TD>
  </TR>
%endif

%if ($(V_mttaxname2) != "")
<TR> <TD ALIGN=right > $(V_mttaxname2) ($(V_mttaxrate2)%) </TD>
  <TD ALIGN=right > $(V_oytax2) $(V_oycpcur)</TD>
 </TR>
%endif

%if ($(V_mttaxname3) != "")
<TR> <TD ALIGN=right > $(V_mttaxname3) ($(V_mttaxrate3)%) </TD>
     <TD ALIGN=right > $(V_oytax3) $(V_oycpcur)</TD>
</TR>
%endif

%if ($(V_mttaxname4) != "")
<TR> <TD ALIGN=right > $(V_mttaxname4) ($(V_mttaxrate4)%) </TD>
  <TD ALIGN=right > $(V_oytax4) $(V_oycpcur)</TD>
</TR>
%endif

%if ($(V_mttaxname5) != "")
<TR> <TD ALIGN=right > $(V_mttaxname5) ($(V_mttaxrate5)%) </TD>
  <TD ALIGN=right > $(V_oytax5) $(V_oycpcur)</TD>
</TR>
%endif

%if ($(V_mttaxname6) != "")
<TR> <TD ALIGN=right > $(V_mttaxname6) ($(V_mttaxrate6)%) </TD>
  <TD ALIGN=right > $(V_oytax6) $(V_oycpcur)</TD>
</TR>
%endif

<TR>
<TD ALIGN=right > <B>Total Sales Tax ($(V_fmttax)%)</B> </TD>
<TD ALIGN=right > <B> $(V_foytax) $(V_oycpcur)</B></TD>
</TR>

<TR>
      <TD ALIGN=right > <B>Shipping Charges</B> </TD>
<TD ALIGN=right > <B> $(V_foyshtot) $(V_oycpcur)</B></TD>
         <TD>
  <select name="shipmode_rn">
           %IF (SHIPPER1 != "" && SHIPPING_NUM != "1" && V_oyshtot == SHCOST1)
<option SELECTED value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
%ELIF (SHIPPER1 != "" && SHIPPING_NUM != "1")
<option value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
  %ENDIF
  %IF (SHIPPER2 != "" && V_oyshtot == SHCOST2)
<option SELECTED value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ELIF (SHIPPER2 != "")
<option value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ENDIF
        %IF (SHIPPING_NUM == "3" && V_oyshtot == SHCOST3)
<option SELECTED value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ELIF (SHIPPING_NUM == "3")
<option value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ENDIF
</SELECT>
</TD> <TD>
<input type=submit value="Update">
</TD>
   </TR>
      <TR><TD></TD></TR>
   <TR> <TD ALIGN=right > <B>TOTAL</B> </TD>
  <TD ALIGN=right > <B> $(V_subtot1) $(V_oycpcur)</B> </TD>
</TR>
%}
</TABLE>
</FORM>
   %}
%MESSAGE{
100: {@DTW_ASSIGN(MERCHANT_TAX, "FALSE")%}:CONTINUE
default: {SHOW_SUBTOTAL_PRICE() not working%}
```

```
%}
%}


%function(dtw_odbc) SHOW_SUBTOTAL_PRICE_MallTax() {
select distinct sanick, salname, safname, oyprtot, oyshtot, oycpcur,
 mhtaxrate, mhtaxrate2, mhtaxrate3, mhtaxrate4, mhtaxrate5, mhtaxrate6,
 oytax1,oytax2, oytax3, oytax4, oytax5, oytax6,
 (mhtaxrate+mhtaxrate2+mhtaxrate3+mhtaxrate4+mhtaxrate5+mhtaxrate6) as mttax,
          (oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as oytax, oysanbr,
          (oyprtot+oyshtot+oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as subtot1
from orderpay, shaddr, shopper, mall
where shopper.shlogid='$(SESSION_ID)' and orderpay.oyornbr=$(order_rn) and
        shaddr.sashnbr=shopper.shrfnbr and shaddr.sarfnbr=orderpay.oysanbr
order by sanick
b%REPORT{
<FORM ACTION="/cgi-bin/ncommerce3/OrderShippingUpdate">
      <INPUT TYPE=hidden NAME=order_rn VALUE=$(order_rn)>
      <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderDisplay?status=P&merchant_rn=$(MerchantRefNum)">
<TABLE  BORDER=0 CELLSPACING=1 CELLPADDING=1 WIDTH=400 ALIGN="left" COLS=3 >
<TR> <TD width=200> </TD>
  <TD width=80> </TD>
  <TD width=120> </TD>
  </TR>
%ROW {
@DTW_format(V_oyshtot, "", "2", V_foyshtot)
@DTW_format(V_oytax,  "", "2", V_foytax)
@DTW_format(V_mttax,  "", "2", V_fmttax)
<TR> <TD ALIGN=right > <B>Sub Total</B> </TD>
  <TD ALIGN=right > $(V_oyprtot) $(V_oycpcur)</TD>
</TR>
%if ($(V_mhtaxrate) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 1 ($(V_mhtaxrate)%) </TD>
  <TD ALIGN=right > $(V_oytax1) $(V_oycpcur)</TD>
  </TR>
      %endif
%if ($(V_mhtaxrate2) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 2 ($(V_mhtaxrate2)%) </TD>
     <TD ALIGN=right > $(V_oytax2) $(V_oycpcur)</TD>
     </TR>
%endif
%if ($(V_mhtaxrate3) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 3 ($(V_mhtaxrate3)%)</TD>
     <TD ALIGN=right > $(V_oytax3) $(V_oycpcur)</TD>
 </TR>
%endif
%if ($(V_mhtaxrate4) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 4 ($(V_mhtaxrate4)%)</TD>
  <TD ALIGN=right > $(V_oytax4) $(V_oycpcur)</TD>
</TR>
%endif
%if  ($(V_mhtaxrate5) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 5 ($(V_mhtaxrate5)%)</TD>
  <TD ALIGN=right > $(V_oytax5) $(V_oycpcur)</TD>
 </TR>
%endif
%if ($(V_mhtaxrate6) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 6 ($(V_mhtaxrate6)%)</TD>
  <TD ALIGN=right > $(V_oytax6) $(V_oycpcur)</TD>
</TR>
%endif
<TR> <TD ALIGN=right ><B>Total Sales Tax ($(V_fmttax)%)</TD>
  <TD ALIGN=right > $(V_foytax) $(V_oycpcur)</B></TD>
</TR>
<TR><TD ALIGN=right ><B>Shipping Charges</TD>
 <TD ALIGN=right >$(V_foyshtot) $(V_oycpcur)</B></TD>
          <TD ALIGN=right >
        <select name="shipmode_rn">
           %IF (SHIPPER1 != "" && SHIPPING_NUM != "1" && V_oyshtot == SHCOST1)
<option SELECTED value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
%ELIF (SHIPPER1 != "" && SHIPPING_NUM != "1")
<option value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
  %ENDIF
  %IF (SHIPPER2 != "" && V_oyshtot == SHCOST2)
<option SELECTED value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ELIF (SHIPPER2 != "")
<option value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
```

```
  %ENDIF
        %IF (SHIPPING_NUM == "3" && V_oyshtot == SHCOST3)
<option SELECTED value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ELIF (SHIPPING_NUM == "3")
<option value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ENDIF
</SELECT>  </TD>
<TD><input type=submit value="Update"></TD>
</TR>
<TR><TD></TD></TR>
<TR><TD ALIGN=right ><B>TOTAL</B></TD>
 <TD ALIGN=right ><B>$(V_subtot1) $(V_oycpcur)</B></TD>
</TR>
   %}
</TABLE>
</FORM>
    %}
%MESSAGE{
default: {SHOW_SUBTOTAL_PRICE_MallTax() not working%}
%}
%}
%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT {
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>$(LongStoreName) Order Information</TITLE>
</HEAD>
<HTML>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
<H2><FONT FACE="helvetica">$(LongStoreName) Order Information</FONT></H2>
<TABLE WIDTH=600 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR><TD><FONT SIZE=2>
The total charges for the items in the Order List are specified below.<BR>
Once you are satisfied with the charges, enter the : <BR>
<ul>
<li>Shipping address that this order and invoice should be sent to.</li>
<li>Credit card information.</li>
</ul>
</TD>
</TR>
</TABLE>
</FONT>
@DETERMINE_SHIPPING_LIST()
<TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR><TD COLSPAN=2><HR WIDTH=550 ALIGN=left></TD> </TR>
<TR><TD><H3><FONT FACE="helvetica">Total Charges:</FONT></H3></TD>
</TR>
<TR><TD><FONT SIZE=2>
Select the shipping service that you would like your purchase to be shipped
with and then update the charge calculation by clicking the update button. </FONT></TD>
</TR>
<TR><TD><BR></TD></TR>
<TR><TD>
@SHOW_SUBTOTAL_PRICE_MerchantTax()
%IF (MERCHANT_TAX == "FALSE")
 @SHOW_SUBTOTAL_PRICE_MallTax()
%ENDIF
</TD></TR>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD><H3><FONT FACE="helvetica">Shipping Address:</FONT></H3></TD></TR>
<TR><TD><FONT SIZE=2>Enter the shipping address for this purchase.  An invoice will also
be sent to this address.</FONT></TD></TR>
<TR><TD><BR></TD></TR>
<TR><FORM ACTION="/cgi-bin/ncommerce3/AddressUpdate" Method="post"><TD>
b
</TD></TR>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD><H3><FONT FACE="helvetica">Payment Information:</FONT></H3></TD></TR>
<TR><TD><FONT SIZE=2><H4>In our shop you have zwo ways to do the payment.</H4><BR> <BR>
You can enter your credit card information in the following feelds, than
your information will be send to us in an encryption way.<BR>
For the more secure way you have the possibiltiy to do your payment <BR>
with your payment wallet via the SET protocoll.<BR></FONT></TD></TR>
<TR><TD><BR></TD></TR>
<TR><TD><TABLE WIDTH=200 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=8 >
```

```
<TR>
%if (CC_visa == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=VISA></TD>
<TD VALIGN="top"><img src="/sggifs/visa.gif"></TD>
%endif
%if (CC_master == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=MAST></TD>
<TD VALIGN="top"><img src="/sggifs/mc.gif"></TD>
%endif
%if(CC_amex == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=AMEX></TD>
<TD VALIGN="top"><img src="/sggifs/amex.gif"></TD>
%endif
%if(CC_discover == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=DISC></TD>
<TD VALIGN="top"><img src="/sggifs/discover.gif"></TD>
%endif
</TR>
</TABLE>
</TD></TR>
<TR><TD><TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=4>
<TR><TD><BR></TD></TR>
<TR><TD ALIGN="left"><FONT SIZE=2>Card Number</FONT></TD>
 <TD ALIGN="left"><FONT SIZE=2>Expiration Month</FONT></TD>
 <TD ALIGN="left"><FONT SIZE=2>Expiration Year</FONT></TD>
</TR>
<TR><TD ALIGN="left" VALIGN=middle><INPUT TYPE=text SIZE=15 MAXLENGTH=256 NAME="ccnum"
VALUE="$(ccnum)"></TD>
    <TD ALIGN="left" VALIGN=middle>
<select name="ccxmonth" size=1>&nbsp&nbsp
<option selected></option>
<option value="1">January</option>
<option value="2">February</option>
<option value="3">March</option>
<option value="4">April</option>
<option value="5">May</option>
<option value="6">June</option>
<option value="7">July</option>
<option value="8">August</option>
<option value="9">September</option>
<option value="10">October</option>
<option value="11">November</option>
<option value="12">December</option>
</select></td>
     <TD align="left" valign=middle>
<select name="ccxyear" size=1>&nbsp&nbsp
<option selected></option>
<option value="1998">1998 </option>
<option value="1999">1999</option>
<option value="2000">2000</option>
<option value="2001">2001</option>
<option value="2002">2002</option>
<option value="2003">2003</option>
<option value="2003">2004</option>
 </SELECT></TD>
</TR></TABLE>
</TD></TR>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD ALIGN="center">
<INPUT TYPE=hidden NAME=sanick VALUE=$(SESSION_ID)>
<INPUT TYPE=hidden NAME=order_rn VALUE=$(order_rn)>
<INPUT Type = "hidden" Name="merchant_rn" Value="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="url"
VALUE="/cgi-bin/ncommerce3/OrderProcess?merchant_rn=$(MerchantRefNum)">
<input type=submit value="Purchase"> </TD></TR>
</FORM>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD ALIGN="center">
<FORM action="/cgi-bin/ncommerce3/pay_wakeup">
<INPUT TYPE=hidden NAME="order_rn" value="$(order_rn)">
<INPUT TYPE=hidden NAME="merchant_rn" value="$(MerchantRefNum)">
   <input type=submit value="Pay with my Wallet for order number: $(order_rn)">
</FORM></TD></TR>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.7 Macro for Accepted the Order (Alternative 2)

This macro, named order.withMerch.Orig.Payment (order.d2w), is nearly the same as the order.d2w we mentioned earlier, but works with merchant originated payment. It also works with the SET protocol.

```
%include "ShopITSO/ShopITSO.inc"
%{======================================================================

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-D245798-NC3

(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

======================================================================%}
%define {
 SHOWSQL="NO"
 MERCHANT_TAX="TRUE"
 SHIPPING_NUM = "0"
 SHIPPER1_RNBR = ""
 SHIPPER2_RNBR = ""
 SHIPPER3_RNBR = ""
 SHIPPER1 = ""
 SHIPPER2 = ""
 SHIPPER3 = ""

%}

%function(dtw_odbc) GET_SHIPPING_ADDRESS_INFO() {
select sarfnbr, safname, salname, saaddr1, saaddr2, sacity, sazipc, sacntry, sastate,
saphone1, saemail1
from shaddr, shipto
where sanick='$(SESSION_ID)' and saadrflg='P' and STSANBR=sarfnbr
%report {
 %ROW{
       @DTW_assign(SHOPPER_REF, V_sarfnbr)
 @DTW_assign(save_sarfnbr, V_sarfnbr)
 @DTW_assign(save_safname, V_safname)
 @DTW_assign(save_salname, V_salname)
 @DTW_assign(save_saaddr1, V_saaddr1)
 @DTW_assign(save_saaddr2, V_saaddr2)
 @DTW_assign(save_sacity, V_sacity)
 @DTW_assign(save_sastate, V_sastate)
 @DTW_assign(save_sazipc, V_sazipc)
 @DTW_assign(save_sacntry, V_sacntry)
 @DTW_assign(save_saphone1, V_saphone1)
 @DTW_assign(save_safax, V_safax)
 @DTW_assign(save_email1, V_email1)
 %}
<INPUT TYPE=hidden NAME="sarfnbr" VALUE="$(save_sarfnbr)">
   <TABLE WIDTH=400 CELLPADDING=0  CELLSPACING=0 BORDER=0>
   <TR>
  <TD ALIGN="left"><FONT SIZE=2>First Name</FONT></TD>
<TD ALIGN="left"><FONT SIZE=2><b>Last Name</b></FONT></TD>
</TR>
   <TR>
<TD COLSPAN=1><INPUT TYPE="text" NAME="safname" VALUE="$(save_safname)" VALUESIZE="28"
MAXLENGTH="30"></TD>
<TD COLSPAN=1><INPUT TYPE="text" NAME="salname" VALUE="$(save_salname)" SIZE="28"
MAXLENGTH="30"></TD>
</TR>
   <TR> <TD ALIGN="left"><FONT SIZE=2><b>Address</b></FONT></TD> </TR>
<TR> <TD COLSPAN=3><INPUT TYPE="text" NAME="saaddr1" VALUE="$(save_saaddr1)" SIZE="62"
MAXLENGTH="50"></TD> </TR>
<TR> <TD COLSPAN=3><INPUT TYPE="text" NAME="saaddr2" VALUE="$(save_saaddr2)" SIZE="62"
MAXLENGTH="50"></TD> </TR>
   <TR> <TD ALIGN=left><FONT SIZE=2><b>City</FONT></b></TD>
  <TD ALIGN=left><FONT SIZE=2><b>State/Province</FONT></b></TD>
</TR>
   <TR> <TD COLSPAN=1><INPUT TYPE="text" NAME="sacity" VALUE="$(save_sacity)" SIZE="28"
MAXLENGTH="30"></TD>
  <TD COLSPAN=1><INPUT TYPE="text" NAME="sastate" VALUE="$(save_sastate)" SIZE="28"
MAXLENGTH="20"></TD>
</TR>
   <TR> <TD ALIGN=left><FONT SIZE=2><b>ZIP/Postal code</b></FONT></TD>
  <TD ALIGN=left><FONT SIZE=2><b>Country</b></FONT></TD>
</TR>
   <TR> <TD COLSPAN=1><INPUT TYPE="text" NAME="sazipc" VALUE="$(save_sazipc)" SIZE="28"
MAXLENGTH="20"></TD>
  <TD COLSPAN=1><INPUT TYPE="text" NAME="sacntry" VALUE="$(save_sacntry)" SIZE="28"
MAXLENGTH="30"></TD>
</TR>
   <TR> <TD ALIGN=left><FONT SIZE=2>Phone Number</FONT></TD>
  <TD ALIGN=left><FONT SIZE=2>E-mail Address</FONT></TD>
</TR>
   <TR> <TD COLSPAN=1><INPUT TYPE="text" NAME="saphone1" VALUE="$(save_saphone1)" SIZE="28"
MAXLENGTH="30"></TD>
  <TD COLSPAN=3><INPUT TYPE="text" NAME="saemail1" VALUE="$(save_saemail1)" SIZE="28"
MAXLENGTH="254"></TD>
</TR>
   </TABLE>
%}
%MESSAGE{
   100: { No Address Found!<p>
   %}
   default: {
<font size=+3><b>Database Error:</b></font><BR>
A database error occurred. Please contact the merchant server administrator.<BR>
SQL error code = $(SQL_CODE)
   %}
%}
%}

%function(dtw_odbc) DETERMINE_SHIPPING_LIST() {
select SPRFNBR, SPCHRGE, SMRFNBR , SMCARRID
fromSHIPPING, SHIPMODE
whereSPMENBR=$(MerchantRefNum) and SPMMNBR =SMRFNBR
order by spchrge ASC
%REPORT{
%ROW{
  @DTW_ASSIGN(SHIPPING_NUM, ROW_NUM)
%IF (ROW_NUM == "1")
@DTW_ASSIGN(SHCOST1, V_SPCHRGE)
@DTW_CONCAT(" -- ", V_SPCHRGE, COST1)
@DTW_CONCAT(V_SMCARRID, COST1, SHIPPER1)
@DTW_ASSIGN(SHIPPER1_RNBR, V_SMRFNBR)
%ENDIF
%IF (ROW_NUM == "2")
@DTW_ASSIGN(SHCOST2, V_SPCHRGE)
@DTW_CONCAT(" -- ", V_SPCHRGE, COST2)
@DTW_CONCAT(V_SMCARRID, COST2, SHIPPER2)
@DTW_ASSIGN(SHIPPER2_RNBR, V_SMRFNBR)
%ENDIF
%IF (SHIPPING_NUM == "3")
@DTW_ASSIGN(SHCOST3, V_SPCHRGE)
@DTW_CONCAT(" -- ", V_SPCHRGE, COST3)
@DTW_CONCAT(V_SMCARRID, COST3, SHIPPER3)
@DTW_ASSIGN(SHIPPER3_RNBR, V_SMRFNBR)
%ENDIF
%}
%}
%MESSAGE{
default: {ERROR : Problem with DISPLAY_SHIPPING_LIST function %}
%}
%}


%function(dtw_odbc) SHOW_SUBTOTAL_PRICE_MerchantTax() {
select distinct sanick, salname, safname, oyprtot, oyshtot, oycpcur,
mttaxrate1, mttaxname1, mttaxrate2, mttaxname2, mttaxrate3, mttaxname3,
mttaxrate4, mttaxname4, mttaxrate5, mttaxname5, mttaxrate6, mttaxname6, mtmenbr,
oytax1,oytax2, oytax3, oytax4, oytax5, oytax6,
(mttaxrate1+mttaxrate2+mttaxrate3+mttaxrate4+mttaxrate5+mttaxrate6) as mttax,
         (oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as oytax, oysanbr,
         (oyprtot+oyshtot+oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as subtot1
from orderpay, shaddr, shopper, merchanttax
where shopper.shlogid='$(SESSION_ID)' and orderpay.oyornbr=$(order_rn) and
         shaddr.sashnbr=shopper.shrfnbr and shaddr.sarfnbr=orderpay.oysanbr
   and mtmenbr=$(MerchantRefNum)
order by sanick

%REPORT{
   <FORM ACTION="/cgi-bin/ncommerce3/OrderShippingUpdate">
      <INPUT TYPE=hidden NAME=order_rn VALUE= $(order_rn)>
      <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderDisplay?status=P&merchant_rn=$(MerchantRefNum)">
      <TABLE  BORDER=0 CELLSPACING=1 CELLPADDING=1 WIDTH=400 ALIGN="left" COLS=3 >
      <TR>
<TD width=200> </TD>
<TD width=180>  </TD>
<TD width=120> </TD>
  </TR>
%ROW {
@DTW_format(V_oyshtot, "", "2", V_foyshtot)
@DTW_format(V_oytax,  "", "2", V_foytax)
@DTW_format(V_mttax,  "", "2", V_fmttax)
      <TR> <TD ALIGN=right > <B>Sub Total</B> </TD>
  <TD ALIGN=right > <B> $(V_oyprtot) $(V_oycpcur)</B></TD>
   </TR>

%if ($(V_mttaxname1) != "")
      <TR> <TD ALIGN=right > $(V_mttaxname1) ($(V_mttaxrate1)%) </TD>
  <TD ALIGN=right > $(V_oytax1) $(V_oycpcur)</TD>
  </TR>
%endif

%if ($(V_mttaxname2) != "")
<TR> <TD ALIGN=right > $(V_mttaxname2) ($(V_mttaxrate2)%) </TD>
  <TD ALIGN=right > $(V_oytax2) $(V_oycpcur)</TD>
 </TR>
%endif

%if ($(V_mttaxname3) != "")
<TR> <TD ALIGN=right > $(V_mttaxname3) ($(V_mttaxrate3)%) </TD>
     <TD ALIGN=right > $(V_oytax3) $(V_oycpcur)</TD>
</TR>
%endif

%if ($(V_mttaxname4) != "")
<TR> <TD ALIGN=right > $(V_mttaxname4) ($(V_mttaxrate4)%) </TD>
  <TD ALIGN=right > $(V_oytax4) $(V_oycpcur)</TD>
</TR>
%endif

%if ($(V_mttaxname5) != "")
<TR> <TD ALIGN=right > $(V_mttaxname5) ($(V_mttaxrate5)%) </TD>
  <TD ALIGN=right > $(V_oytax5) $(V_oycpcur)</TD>
</TR>
%endif

%if ($(V_mttaxname6) != "")
<TR> <TD ALIGN=right > $(V_mttaxname6) ($(V_mttaxrate6)%) </TD>
  <TD ALIGN=right > $(V_oytax6) $(V_oycpcur)</TD>
</TR>
%endif

<TR>
<TD ALIGN=right > <B>Total Sales Tax ($(V_fmttax)%)</B> </TD>
<TD ALIGN=right > <B> $(V_foytax) $(V_oycpcur)</B></TD>
</TR>

<TR>
      <TD ALIGN=right > <B>Shipping Charges</B> </TD>
<TD ALIGN=right > <B> $(V_foyshtot) $(V_oycpcur)</B></TD>
         <TD>
  <select name="shipmode_rn">
           %IF (SHIPPER1 != "" && SHIPPING_NUM != "1" && V_oyshtot == SHCOST1)
<option SELECTED value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
```

```
%ELIF (SHIPPER1 != "" && SHIPPING_NUM != "1")
<option value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
  %ENDIF
  %IF (SHIPPER2 != "" && V_oyshtot == SHCOST2)
<option SELECTED value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ELIF (SHIPPER2 != "")
<option value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ENDIF
        %IF (SHIPPING_NUM == "3" && V_oyshtot == SHCOST3)
<option SELECTED value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ELIF (SHIPPING_NUM == "3")
<option value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ENDIF
</SELECT>
</TD> <TD>
<input type=submit value="Update">
</TD>
   </TR>
      <TR><TD></TD></TR>
   <TR> <TD ALIGN=right > <B>TOTAL</B> </TD>
  <TD ALIGN=right > <B> $(V_subtot1) $(V_oycpcur)</B> </TD>
</TR>
%}
</TABLE>
</FORM>
   %}
%MESSAGE{
100: {@DTW_ASSIGN(MERCHANT_TAX, "FALSE")%}:CONTINUE
default: {SHOW_SUBTOTAL_PRICE() not working%}
%}
%}


%function(dtw_odbc) SHOW_SUBTOTAL_PRICE_MallTax() {
select distinct sanick, salname, safname, oyprtot, oyshtot, oycpcur,
 mhtaxrate, mhtaxrate2, mhtaxrate3, mhtaxrate4, mhtaxrate5, mhtaxrate6,
 oytax1,oytax2, oytax3, oytax4, oytax5, oytax6,
 (mhtaxrate+mhtaxrate2+mhtaxrate3+mhtaxrate4+mhtaxrate5+mhtaxrate6) as mttax,
          (oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as oytax, oysanbr,
          (oyprtot+oyshtot+oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as subtot1
from orderpay, shaddr, shopper, mall
where shopper.shlogid='$(SESSION_ID)' and orderpay.oyornbr=$(order_rn) and
        shaddr.sashnbr=shopper.shrfnbr and shaddr.sarfnbr=orderpay.oysanbr
order by sanick
%REPORT{
<FORM ACTION="/cgi-bin/ncommerce3/OrderShippingUpdate">
      <INPUT TYPE=hidden NAME=order_rn VALUE=$(order_rn)>
      <INPUT TYPE=hidden NAME=url
VALUE="/cgi-bin/ncommerce3/OrderDisplay?status=P&merchant_rn=$(MerchantRefNum)">
<TABLE  BORDER=0 CELLSPACING=1 CELLPADDING=1 WIDTH=400 ALIGN="left" COLS=3 >
<TR> <TD width=200> </TD>
  <TD width=80> </TD>
  <TD width=120> </TD>
  </TR>
%ROW {
@DTW_format(V_oyshtot, "", "2", V_foyshtot)
@DTW_format(V_oytax,  "", "2", V_foytax)
@DTW_format(V_mttax,  "", "2", V_fmttax)
<TR> <TD ALIGN=right > <B>Sub Total</B> </TD>
  <TD ALIGN=right > $(V_oyprtot) $(V_oycpcur)</TD>
</TR>
%if ($(V_mhtaxrate) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 1 ($(V_mhtaxrate)%) </TD>
  <TD ALIGN=right > $(V_oytax1) $(V_oycpcur)</TD>
  </TR>
      %endif
%if ($(V_mhtaxrate2) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 2 ($(V_mhtaxrate2)%) </TD>
     <TD ALIGN=right > $(V_oytax2) $(V_oycpcur)</TD>
     </TR>
%endif
%if ($(V_mhtaxrate3) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 3 ($(V_mhtaxrate3)%)</TD>
     <TD ALIGN=right > $(V_oytax3) $(V_oycpcur)</TD>
 </TR>
%endif
%if ($(V_mhtaxrate4) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 4 ($(V_mhtaxrate4)%)</TD>
```

```
  <TD ALIGN=right > $(V_oytax4) $(V_oycpcur)</TD>
</TR>
%endif
%if  ($(V_mhtaxrate5) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 5 ($(V_mhtaxrate5)%)</TD>
  <TD ALIGN=right > $(V_oytax5) $(V_oycpcur)</TD>
 </TR>
%endif
%if ($(V_mhtaxrate6) != "0.00")
<TR> <TD ALIGN=right > Tax Rate 6 ($(V_mhtaxrate6)%)</TD>
  <TD ALIGN=right > $(V_oytax6) $(V_oycpcur)</TD>
</TR>
%endif
<TR> <TD ALIGN=right ><B>Total Sales Tax ($(V_fmttax)%)</TD>
  <TD ALIGN=right > $(V_foytax) $(V_oycpcur)</B></TD>
</TR>
<TR><TD ALIGN=right ><B>Shipping Charges</TD>
 <TD ALIGN=right >$(V_foyshtot) $(V_oycpcur)</B></TD>
          <TD ALIGN=right >
        <select name="shipmode_rn">
           %IF (SHIPPER1 != "" && SHIPPING_NUM != "1" && V_oyshtot == SHCOST1)
<option SELECTED value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
%ELIF (SHIPPER1 != "" && SHIPPING_NUM != "1")
<option value="$(SHIPPER1_RNBR)"><FONT SIZE=2>$(SHIPPER1)</FONT> </option>
  %ENDIF
  %IF (SHIPPER2 != "" && V_oyshtot == SHCOST2)
<option SELECTED value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ELIF (SHIPPER2 != "")
<option value="$(SHIPPER2_RNBR)"><FONT SIZE=2>$(SHIPPER2)</FONT></option>
  %ENDIF
        %IF (SHIPPING_NUM == "3" && V_oyshtot == SHCOST3)
<option SELECTED value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ELIF (SHIPPING_NUM == "3")
<option value="$(SHIPPER3_RNBR)"><FONT SIZE=2>$(SHIPPER3)</FONT></option>
  %ENDIF
</SELECT>  </TD>
<TD><input type=submit value="Update"></TD>
</TR>
<TR><TD></TD></TR>
<TR><TD ALIGN=right ><B>TOTAL</B></TD>
 <TD ALIGN=right ><B>$(V_subtot1) $(V_oycpcur)</B></TD>
</TR>
   %}
</TABLE>
</FORM>
    %}
%MESSAGE{
default: {SHOW_SUBTOTAL_PRICE_MallTax() not working%}
%}
%}
%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT {
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>$(LongStoreName) Order Information</TITLE>
</HEAD>
<HTML>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
<H2><FONT FACE="helvetica">$(LongStoreName) Order Information</FONT></H2>
<TABLE WIDTH=600 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR><TD><FONT SIZE=2>
The total charges for the items in the Order List are specified below.<BR>
Once you are satisfied with the charges, enter the : <BR>
<ul>
<li>Shipping address that this order and invoice should be sent to.</li>
<li>Credit card information.</li>
</ul>
</TD>
</TR>
</TABLE>
</FONT>
@DETERMINE_SHIPPING_LIST()
<TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR><TD COLSPAN=2><HR WIDTH=550 ALIGN=left></TD> </TR>
<TR><TD><H3><FONT FACE="helvetica">Total Charges:</FONT></H3></TD>
```

```
</TR>
<TR><TD><FONT SIZE=2>
Select the shipping service that you would like your purchase to be shipped
with and then update the charge calculation by clicking the update button. </FONT></TD>
</TR>
<TR><TD><BR></TD></TR>
<TR><TD>
@SHOW_SUBTOTAL_PRICE_MerchantTax()
%IF (MERCHANT_TAX == "FALSE")
 @SHOW_SUBTOTAL_PRICE_MallTax()
%ENDIF
</TD></TR>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD><H3><FONT FACE="helvetica">Shipping Address:</FONT></H3></TD></TR>
<TR><TD><FONT SIZE=2>Enter the shipping address for this purchase.  An invoice will also
be sent to this address.</FONT></TD></TR>
<TR><TD><BR></TD></TR>
<TR><FORM ACTION="/cgi-bin/ncommerce3/AddressUpdate" Method="post"><TD>
@GET_SHIPPING_ADDRESS_INFO()
</TD></TR>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD><H3><FONT FACE="helvetica">Payment Information:</FONT></H3></TD></TR>
<TR><TD><FONT SIZE=2><H4>In our shop you have two ways to do the payment.</H4><BR> <BR>
You can enter your credit card information in the following feelds, than
your information will be send to us in an encryption way.<BR>
For the more secure way you have the possibiltiy to do your payment <BR>
with your payment wallet via the SET protocoll.<BR></FONT></TD></TR>
<TR><TD><BR></TD></TR>
<TR><TD><TABLE WIDTH=200 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=8 >
<TR>
%if (CC_visa == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=VISA></TD>
<TD VALIGN="top"><img src="/sggifs/visa.gif"></TD>
%endif
%if (CC_master == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=MAST></TD>
<TD VALIGN="top"><img src="/sggifs/mc.gif"></TD>
%endif
%if(CC_amex == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=AMEX></TD>
<TD VALIGN="top"><img src="/sggifs/amex.gif"></TD>
%endif
%if(CC_discover == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=DISC></TD>
<TD VALIGN="top"><img src="/sggifs/discover.gif"></TD>
%endif
</TR>
</TABLE>
</TD></TR>
<TR><TD><TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=4>
<TR><TD><BR></TD></TR>
<TR><TD ALIGN="left"><FONT SIZE=2>Card Number</FONT></TD>
 <TD ALIGN="left"><FONT SIZE=2>Expiration Month</FONT></TD>
 <TD ALIGN="left"><FONT SIZE=2>Expiration Year</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>&nbsp&nbsp
<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="2003">2004</option>
 </SELECT></TD>
</TR></TABLE>
</TD></TR>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD ALIGN="center">
<INPUT TYPE=hidden NAME=sanick VALUE=$(SESSION_ID)>
<INPUT TYPE=hidden NAME=order_rn VALUE=$(order_rn)>
<INPUT Type = "hidden" Name="merchant_rn" Value="$(MerchantRefNum)">
<INPUT TYPE=hidden NAME="url"
VALUE="/cgi-bin/ncommerce3/pay_accept?merchant_rn=$(MerchantRefNum)&order_rn=$(order_rn)">
<input type=submit value="Purchase"> </TD></TR>
</FORM>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD ALIGN="center">
<FORM action="/cgi-bin/ncommerce3/pay_wakeup">
<INPUT TYPE=hidden NAME="order_rn" value="$(order_rn)">
<INPUT TYPE=hidden NAME="merchant_rn" value="$(MerchantRefNum)">
   <input type=submit value="Pay with my Wallet for: $(order_rn)">
</FORM></TD></TR>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.8 Macro for Order Confirmation

The Net.Data macro orderok.d2w builds our confirmation page:

```
%include "ShopITSO/ShopITSO.inc"
%{=====================================================================
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-D245798-NC3

(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
=====================================================================%}
%define {
    SHOWSQL="NO"
    MERCHANT_TAX="TRUE"
    SHOPPER_REF = ""
    SHADDR_REF = ""
    ADDR1 = ""
    ADDR2 = ""
    ADDR3 = ""
    ADDR4 = ""
    ADDR5 = ""
    ADDR6 = ""
    ADDR7 = ""
    ADDR8 = ""
    ADDR9 = ""
%}

%{==== GET_SHOPPER_REF_NUM Function ====%}
%function(dtw_odbc) GET_SHOPPER_REF_NUM() {
   select shrfnbr from shopper where shlogid = '$(SESSION_ID)'
   %REPORT{
     %ROW{
       @DTW_assign(SHOPPER_REF, V_shrfnbr)
```

```
      %}
    %}
    %MESSAGE{
      default: { ERROR in GET_SHOPPER_REF_NUM %}
    %}
%}


%function(dtw_odbc) CLEANUP_SHIPTO_ADDRESS() {
    UPDATE shipto SET stsanbr=$(SHADDR_REF) WHERE stornbr=$(order_rn)
    %REPORT{
      %ROW{
      %}
    %}
    %MESSAGE{
      default: { Error updating Shipto table%}
    %}
%}

%function(dtw_odbc) CLEANUP_ORDERPAY_ADDRESS() {
    UPDATE orderpay SET oysanbr=$(SHADDR_REF) WHERE oyornbr=$(order_rn)
    %REPORT{
      %ROW{
      %}
    %}
    %MESSAGE{
      default: { Error updating Orderpay table%}
    %}
%}

%function(dtw_odbc) SHOPPER_INFO() {
select sarfnbr, salname, safname, saaddr1, saaddr2, sacity, sastate, sazipc, sacntry,
saemail1
from shaddr
where sanick='$(SESSION_ID)' and saadrflg='P'
    %REPORT{
      %ROW{
@DTW_assign(SHADDR_REF, V_sarfnbr)
@DTW_assign(ADDR1, V_safname)
@DTW_assign(ADDR2, V_salname)
@DTW_assign(ADDR3, V_saaddr1)
@DTW_assign(ADDR4, V_saaddr2)
@DTW_assign(ADDR5, V_sacity)
@DTW_assign(ADDR6, V_sastate)
@DTW_assign(ADDR7, V_sazipc)
@DTW_assign(ADDR8, V_sacntry)
@DTW_assign(ADDR9, V_saemail1)
 %}
       %}
    %MESSAGE{default: { ERROR in SHOPPER_INFO %}
%}
%}


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

%REPORT{
<BR>
<TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR> <TD COLSPAN=3>
<B><FONT SIZE=2>The following items were purchased:</FONT></B> </TD> </TR>
</TABLE>
<BR>
<TABLE WIDTH=500 CELLPADDING=2 CELLSPACING=0 BORDER=1>
<TR>
    <TD bgcolor=$(BodyColor2)><B><FONT SIZE=2>Product Number</FONT></B></TD>
 <TD bgcolor=$(BodyColor2)><B><FONT SIZE=2>Product Name</FONT></B></TD>
             <TD bgcolor=$(BodyColor2)><B><FONT SIZE=2>QTY you ordered</FONT></B></TD>
    <TD bgcolor=$(BodyColor2) width=20><B><FONT SIZE=2>Product Price</FONT></B></TD>
    <TD bgcolor=$(BodyColor2)><B><FONT SIZE=2>Total Price</FONT></B></TD>
</TR>
%ROW{
     @DTW_MULTIPLY(V_stquant, V_stprice, SUB_TOT)
<TR>
    <TD BGCOLOR="white"><FONT SIZE=2>$(V_prnbr)</FONT></TD>
 <TD BGCOLOR="white"><FONT SIZE=2> $(V_prsdesc)</FONT></TD>
             <TD BGCOLOR="white" ALIGN="right"><FONT SIZE=2> $(V_stquant)</FONT></TD>
    <TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2> $(V_stprice) $(V_stcpcur)</FONT></TD>
    <TD align="right" BGCOLOR="white"><FONT Size=2> $(SUB_TOT) $(V_stcpcur)</FONT></TD>
</TR>
<TR><TD HEIGHT=5></TD></TR>
%}
%}
%MESSAGE{
100    : {<BR><FONT SIZE=2><B>Your Order List is empty.</B></FONT>%}:continue
default: {ERROR : Problem with DISPLAY_DETAILS_LIST function %}
%}
%}




%function(dtw_odbc) DISPLAY_CHARGES_MallTax() {
select distinct sanick, salname, safname, oyprtot, oyshtot, oycpcur,
mhtaxrate, mhtaxrate2, mhtaxrate3, mhtaxrate4, mhtaxrate5, mhtaxrate6,
oytax1,oytax2, oytax3, oytax4, oytax5, oytax6,
(mhtaxrate+mhtaxrate2+mhtaxrate3+mhtaxrate4+mhtaxrate5+mhtaxrate6) as mttax,
           (oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as oytax, oysanbr,
          (oyprtot+oyshtot+oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as subtot1
from orderpay, shaddr, shopper, mall
where shopper.shlogid='$(SESSION_ID)' and orderpay.oyornbr=$(order_rn) and
        shaddr.sashnbr=shopper.shrfnbr and shaddr.sarfnbr=orderpay.oysanbr
order by sanick
%REPORT{
%ROW{
@DTW_format(V_oyshtot, "", "2", V_foyshtot)
@DTW_format(V_oytax,  "", "2", V_foytax)
@DTW_format(V_mttax,  "", "2", V_fmttax)
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>Subtotal</B></FONT></TD>
<TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2>$(V_oyprtot) $(V_oycpcur)</FONT></TD> </TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>Total Sales Tax ($(V_fmttax)%)</B></FONT></TD>
      <TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2>$(V_foytax)  $(V_oycpcur)</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>Shipping Charge</b></FONT></TD>
<TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2>$(V_foyshtot)  $(V_oycpcur)</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>TOTAL</B></FONT></TD>
         <TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2><B>$(V_subtot1)
$(V_oycpcur)</B></FONT></TD>
</TR>
%}
 </TABLE>
 <BR>
%}
%MESSAGE{
100 : {<BR><FONT SIZE=2><B>Your Order List is empty.</B></FONT>%}:continue
default: { Error %}
%}
%}

%function(dtw_odbc) DISPLAY_CHARGES_MerchantTax() {
select distinct sanick, salname, safname, oyprtot, oyshtot, oycpcur,
mttaxrate1, mttaxname1, mttaxrate2, mttaxname2, mttaxrate3, mttaxname3,
mttaxrate4, mttaxname4, mttaxrate5, mttaxname5, mttaxrate6, mttaxname6, mtmenbr,
oytax1,oytax2, oytax3, oytax4, oytax5, oytax6,
(mttaxrate1+mttaxrate2+mttaxrate3+mttaxrate4+mttaxrate5+mttaxrate6) as mttax,
           (oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as oytax, oysanbr,
          (oyprtot+oyshtot+oytax1+oytax2+oytax3+oytax4+oytax5+oytax6) as subtot1
from orderpay, shaddr, shopper, merchanttax
where shopper.shlogid='$(SESSION_ID)' and orderpay.oyornbr=$(order_rn) and
        shaddr.sashnbr=shopper.shrfnbr and shaddr.sarfnbr=orderpay.oysanbr
and mtmenbr=$(MerchantRefNum)
order by sanick
%REPORT{
%ROW{
@DTW_format(V_oyshtot, "", "2", V_foyshtot)
```

```
@DTW_format(V_oytax,  "", "2", V_foytax)
@DTW_format(V_mttax,  "", "2", V_fmttax)
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>Subtotal</B></FONT></TD>
<TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2>$(V_oyprtot) $(V_oycpcur)</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>Total Sales Tax ($(V_fmttax)%)</B></FONT></TD>
         <TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2>$(V_foytax)  $(V_oycpcur)</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>Shipping Charge</b></FONT></TD>
         <TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2>$(V_foyshtot)
$(V_oycpcur)</FONT></TD>
</TR>
<TR>
<TD ALIGN="right" COLSPAN=4><FONT SIZE=2><B>TOTAL</B></FONT></TD>
         <TD ALIGN="right" BGCOLOR="white"><FONT SIZE=2><B>$(V_subtot1)
$(V_oycpcur)</B></FONT></TD>
</TR>
%}
 </TABLE>
 <BR>
%}
%MESSAGE{
default: {@DTW_ASSIGN(MERCHANT_TAX, "FALSE") %}:CONTINUE
%}
%}


%function (DTW_ODBC) GET_CONTACT_INFO() {
select mename, mestname, meaddr1, mecity, mestate, mezipc, mecntry, mephone, mecmail1
from merchant
where MERFNBR=$(MerchantRefNum)
%report {
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=400>
 <TR><TD><FONT SIZE=2>If you have any problems with this order, please contact us at
:</FONT><BR><BR></TD</TR>
%row {
<TR>
<TD ALIGN="left">
<FONT SIZE=2>
<address>
$(V_mestname)<BR>
$(V_MEADDR1)<br>
$(V_MECITY), $(V_MESTATE)<br>
$(V_MEZIPC), $(V_MECNTRY)<br>
Telephone: $(V_MEPHONE)<br>
Email:  $(V_MECMAIL1)<br>
</address>
</FONT>
</TD>
</TR>
%}
</TABLE>
%}
%MESSAGE{
100: {  No Contact Information Available. %} : continue
default: {
<font size=2>
<b>Database Error:
</b><p>
A database error occurred when attempting to
get Address Information.  Please contact the
merchant server administrator.
  </font>
%}
%}

%}



%{==================================================%}
%{ HTML Report Section
%{==================================================%}
%HTML_REPORT{
<HTML>
```

```
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
<TITLE>$(LongStoreName) Order Confirmation</TITLE>
</HEAD>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
<TABLE WIDTH=600 CELLSPACING=0 CELLPADDING=0 BORDER=0>
<H2><FONT FACE="helvetica">$(LongStoreName) Order Confirmation</FONT></H2>
@GET_SHOPPER_REF_NUM()
@SHOPPER_INFO()
<TR><TD><B><FONT SIZE=3>Thank you $(V_safname) $(V_salname) for shopping at
$(LongStoreName).</FONT></B>
 <BR><BR> </TD> </TR>
<TR><TD WIDTH=600><B><FONT SIZE=2>Your order has been received and is being processed.<BR>
You will be notified when it is confirmed. <br> <br> <br></B></TD></TR>
<TR><TD ><B><FONT SIZE=2>Your customer number is: $(sanick)</TD> </B></TR>
<TR><TD><B><FONT SIZE=2>Your order number is:     $(order_rn)</TD> </B></TR>
<TR><TD COLSPAN=2><FONT SIZE=2><BR>
Please retain the above numbers for your records.  You can check the status of your order at
any time
by clicking on the <A
HREF="/cgi-bin/ncommerce3/OrderList?merchant_rn=$(MerchantRefNum)&status=C"
TARGET="main">Order Status</a> option and providing your customer & order numbers.
   </FONT> </TD> </TR>
<TR> <TD COLSPAN=2> <BR> <FONT SIZE=2> <B>This order (with the invoice) will be shipped to
:</B> <BR>
   $(ADDR1) $(ADDR2)<BR>
$(ADDR3)<br>
$(ADDR4)<br>
$(ADDR5), $(ADDR6)<br>
$(ADDR7), $(ADDR8)<br>
$(ADDR9)
</FONT></TD></TR>
</TABLE>
@CLEANUP_SHIPTO_ADDRESS()
@CLEANUP_ORDERPAY_ADDRESS()
@DISPLAY_DETAILS_LIST()
@DISPLAY_CHARGES_MerchantTax()
%IF( MERCHANT_TAX == "FALSE")
@DISPLAY_CHARGES_MallTax()
%ENDIF
@GET_CONTACT_INFO()
</body>
</html>
%}
```

A.9.9 Macro for Order Status

This macro named orderlstc.d2w displays the order status:

```
%include "ShopITSO/ShopITSO.inc"
%{======================================================================
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-D245798-NC3

(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 "platform.d2w"
%define {
 SHOWSQL="NO"
%}
```

```
%function(dtw_odbc) GET_ORDER_STATUS() {
selectORRFNBR, $(NC_DATE) (orpstmp) as pdate, $(NC_TIME) (orpstmp) as ptime, orstat
from   ORDERS, SHOPPER
whereORRFNBR=$(order_rn) and SHLOGID='$(customer_rn)' and ORMENBR=$(MerchantRefNum)
and shrfnbr=orshnbr
%REPORT {
         <TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD COLSPAN=3><FONT SIZE=2>Status information: </FONT><BR><BR></TD></TR>
%ROW {
<TR><TD align=left><FONT SIZE=2><B>Order Status:</B></font></td>
%if (V_orstat == "C")
<td align=right COLSPAN=2><FONT SIZE=2>"Order accepted and in process"</font></td></TR>
   <TR><TD><BR></TD></TR>
   <TR><TD align=left><FONT SIZE=2><B>Date accepted:</B></font></td>
    <TD align=right COLSPAN=2><FONT SIZE=2>$(V_pdate)</font></td></TR>
   <TR><TD align=left><FONT SIZE=2><B>Time Completed:</B></font></td>
       <TD align=right COLSPAN=2><FONT SIZE=2>&nbsp;$(V_ptime)&nbsp;</font></td></TR>
   %elif (V_orstat == "9")
   <td align=right COLSPAN=2><FONT SIZE=2>"Order shipped"</font></td></TR>
   <TR><TD><BR></TD></TR>
   <TR><TD align=left><FONT SIZE=2><B>Date shipped:</B></font></td>
    <TD align=right COLSPAN=2><FONT SIZE=2>$(V_pdate)</font></td></TR>
   %else
<TR><TD align=right COLSPAN=2><FONT SIZE=2>"Pending"</font></td></TR>
%endif
<TR><TD COLSPAN=3> <HR> </TD></TR>
%}
</table>
</center>
%}
%MESSAGE{
                    100 : {
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD COLSPAN=2><FONT size=2> Sorry! Order status information is not available for
:</FONT><BR><BR></TD></TR>
         <TR><TD><FONT size=2><B>Customer Number</B></FONT></TD>
             <TD><FONT size=2>$(customer_rn)</FONT></TD></TR>
<TR><TD><FONT size=2><B>Order Number</B></font></TD>
 <TD><FONT size=2>$(order_rn)</font></TD></TR>
</TABLE>
          %}
          default: {
         <FONT size=2><B>More information is needed.</B><BR>
<BR>Both a <i><B>customer number</b></I> and an <i><b>order number</b></i>
   is needed to check your order status. Please try again.</font>
          %}
   %}
%}


%function(dtw_odbc) GET_ORDER_DETAILS() {
selectstprnbr, stprice, stcpcur, prsdesc, prnbr
from   SHOPPER, SHIPTO, PRODUCT
wherestornbr=$(order_rn) and stmenbr=$(MerchantRefNum)
and prrfnbr=stprnbr and prmenbr=$(MerchantRefNum)
and shlogid='$(customer_rn)' and shrfnbr=stshnbr
%REPORT {
   <TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
  %ROW {
       <TR><TD COLSPAN=2><FONT SIZE=2>#$(V_prnbr)     : $(V_prsdesc)</FONT></TD>
        <TD align=right><FONT SIZE=2>$(V_stprice) $(V_stcpcur)</FONT></TD></TR>
%}
</table>
</center>
%}
%MESSAGE{
           default: {
            <font size=2><b>Database Error: <I>ORDER/LIST</I></b></font><br>
            A database error occurred. Please contact the merchant server administrator.<BR>
            Error Code = $(SQL_CODE)<BR>
           %}
    %}
%}


%function(dtw_odbc) GET_ORDER_CHARGES() {
selectORRFNBR, ORPRTOT, ORSHTOT, ORTXTOT, ((ORPRTOT+ORSHTOT+ORTXTOT)) as grandtot,
ormenbr, orcpcur
from   ORDERS, SHOPPER
whereORRFNBR=$(order_rn) and SHLOGID='$(customer_rn)' and ORMENBR=$(MerchantRefNum)
and shrfnbr=orshnbr
%REPORT {
   <TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
     %ROW {
       <TR><TD COLSPAN=3><HR></TD></TR>
       <TR><TD align=right><FONT SIZE=2><B>Sub total</B></font></td>
  <TD align=right COLSPAN=2><FONT SIZE=2>$(V_orprtot) $(V_orcpcur)</font></td></TR>
 <TR><TD align=right><FONT SIZE=2><B>Tax</B></font></td>
  <TD align=right COLSPAN=2><FONT SIZE=2>$(V_ortxtot) $(V_orcpcur)</font></td></TR>
 <TR><TD align=right><FONT SIZE=2><B>Shipping Charge</B></font></td>
  <TD align=right COLSPAN=2><FONT SIZE=2>$(V_orshtot) $(V_orcpcur)</font></td></TR>
 <TR><TD align=right><FONT SIZE=2><B>TOTAL CHARGE</B></font></td>
  <TD align=right COLSPAN=2><FONT SIZE=2><B>$(V_grandtot) $(V_orcpcur)</B></font></td></TR>
%}
</table>
</center>
%}
%MESSAGE{
     default: {
       <font size=2><b>Database Error: <I>ORDER/LIST</I></b></font><br>
       A database error occurred. Please contact the merchant server administrator.<BR>
       Error Code = $(SQL_CODE)<BR>
     %}
    %}
%}


%HTML(INPUT) {
<HTML>
<HEAD>
<META HTTP-EQUIV=Expires CONTENT="Mon, 01 Jan 1996 01:01:01 GMT">
</HEAD>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD ALIGN="left" VALIGN="center" COLSPAN=2><FONT COLOR="$(TitleTxtCol)" FACE="helvetica">
     <H3> Order # $(order_rn) Status: </H3><HR><BR>
     <H4> Custmer # $(customer_rn)</h4></FONT></TD></TR>
</TABLE>
@GET_ORDER_STATUS()
@GET_ORDER_DETAILS()
@GET_ORDER_CHARGES()
</BODY>
</HTML>
%}



%HTML(REPORT) {
<HTML>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
<FORM METHOD="POST"
ACTION="/cgi-bin/ncommerce3/ExecMacro/$(STORENAME)/orderlstc.d2w/input">
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD ALIGN="left" VALIGN="center" COLSPAN=2><FONT COLOR="$(TitleTxtCol)" FACE="helvetica">
      <H3>Order Status</H3></FONT></TD></TR>
<TR><TD ALIGN="left" VALIGN="center" COLSPAN=2><FONT SIZE=2>
Please enter your customer number and order number that was displayed when your
order was confirmed. </FONT><BR><BR></TD></TR><TR>
 <TD ALIGN="left"><FONT SIZE=2>Customer #</FONT></TD>
 <TD ALIGN="left"><FONT SIZE=2>Order #</FONT></TD></TR>
<TR><TD ALIGN="left"><INPUT TYPE="text" NAME="customer_rn" SIZE="20" MAXLENGTH="30"></TD>
    <TD ALIGN="leftt"><INPUT TYPE="text" NAME="order_rn" SIZE="5" MAXLENGTH="10"></TD></TR>
<TR><TD ALIGN="center" COLSPAN=2><BR><INPUT TYPE="submit" value="Retrieve Order
Status"></TD></TR>
</TABLE>
</FORM>
</BODY>
</HTML>
%}
```

A.9.10 Macro for Search

The Net.Data macro searchrslt.d2w builds the search result and is invoked by the search.html file:

```
%include "ShopITSO/ShopITSO.inc"
%{========================================================================
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-D245798-NC3

(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
========================================================================%}
%define {
  SHOWSQL = "NO"
ADDRESS_REF = ""
%}

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


%function(dtw_odbc) SEARCH_PRODUCTS() {
  select PRODUCT.PRRFNBR, PRODUCT.PRNBR, PRSDESC   from PRODUCT
  where prpub=1 and prmenbr=$(MerchantRefNum) and (translate(prsdesc) like
'%@DTW_rTRANSLATE($(search))%')
  %REPORT{
 <TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
 <TR><TD ALIGN="left" VALIGN="center">
     <FONT COLOR="$(TitleTxtCol)" FACE="helvetica"><H2>Search Results</H2></FONT></TD></TR>
 <TR><TD ALIGN="left" VALIGN="center" COLSPAN=2>
      <H2>You can find the results of your search for <I><B>$(search)</B></I> below:
</H2></TD></TR>
 <TR><TD><H4>When you select one product, you get more information about it and also you can
order </H4></TD></TR>
    %ROW{
<TR><TD VALIGN="top"><FONT SIZE=2>
<UL><LI>
<A
HREF="/cgi-bin/ncommerce3/ProductDisplay?prrfnbr=$(V_PRRFNBR)&prmenbr=$(MerchantRefNum
)">$(V_PRSDESC)</A>
</LI></UL>
</FONT>
</TD>
      <TD></TD></TR>
    %}
</TABLE>
 %}
 %MESSAGE{100:{
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0 WIDTH=100%>
<TR><TD><h2>No Matches found for <I><B>$(search)</B></I></h2></TD></TR>
</TABLE>
%} :continue %}
%}
```

```
%HTML_REPORT {
<HTML>
<HEAD>
</HEAD>
<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">
@GET_ADDRESS_REF_NUM()
@SEARCH_PRODUCTS()
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=2 WIDTH=50%>
<TR><TD><H4>Try another Search.<H4></TD></tr>
<TR><TD align ="left"><FORM METHOD="POST"
ACTION="/cgi-bin/ncommerce3/ExecMacro/$(STORENAME)/searchrslt.d2w/report">
     <INPUT TYPE="text" NAME="search" SIZE="30" MAXLENGTH="30"></td>
     <TD><INPUT TYPE="submit" value="Search"></TD></TR>
  </FORM>
</TABLE>
</BODY>
</HTML>
%}
```

A.9.11 Error Macro Address Update

The Net.Data macro err_adrbk_up.d2w is shown when the shopper enters address information that is not valid, or the address or credit card information is missing:

```
%include "ShopITSO/ShopITSO.inc"

%{======================================================================

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.      All Rights Reserved

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

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

%define{

sashnbrVal  = sashnbr ?  "$(sashnbr)" :  "null"

l_sanick = "Nick Name <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_satitle = "Title"
l_salname = "Last Name <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_safname = "First Name"
l_saaddr1 = "Address <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_sacity = "City <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_sastate =  "State/Province <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_sazipc = "Zip/Postal Code <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_sacntry = "Country <FONT SIZE=2 COLOR=$(NavLinkCol)>(mandatory)</FONT>"
l_saphone1 = "Daytime Phone Number"
l_saphone2 = "Evening Phone Number"
l_saemail1 = "E-mail Address"


%}

%function(dtw_odbc) CheckRegistered(){
```

```
SELECT satitle, safname, samname, salname, shshtyp
FROM shaddr, shopper
WHERE sashnbr=shrfnbr and sanick='$(SESSION_ID)' and shlogid='$(SESSION_ID)' and
(shshtyp='R' or shshtyp='A') and saadrflg='P'

  %REPORT{
     %ROW{
$(V_safname) $(V_samname) $(V_salname)
     %}
  %}

  %MESSAGE{
     100 : {Guest Shopper

     %} : continue
     default: {
     %} :continue
  %}

%}


%HTML_REPORT{

<HTML>

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

<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">

<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>

<TR>
<TD ALIGN="left" VALIGN="center">
<H2><FONT FACE="helvetica">$(LongStoreName) Order Information</FONT></H2>
</TD>
</TR>

</TABLE>

<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>

<TR>
<TD align=left colspan=3 BGCOLOR="pink">

<FONT SIZE=2><B>
Your Purchase has not been accepted.
There is a problem with the information you provided.

%if (error_code == "160")
  The information in the <i><a href="#$(field)">$(msg)</a></i> field is invalid.
%elif ((error_code == "190") && (field == "salname"))
  The <i><a href="#$(field)">Last Name</a></i> field must be filled in.
%elif ((error_code == "190") && (field == "saaddr1"))
  The <i><a href="#$(field)">Address</a></i> field must be filled in.
%elif ((error_code == "190") && (field == "sacity"))
  The <i><a href="#$(field)">City</a></i> field must be filled in.
%elif ((error_code == "190") && (field == "sastate"))
  The <i><a href="#$(field)">State/Province</a></i> field must be filled in.
%elif ((error_code == "190") && (field == "sacntry"))
  The <i><a href="#$(field)">Country</a></i> field must be filled in.
%elif ((error_code == "190") && (field == "sazipc"))
  The <i><a href="#$(field)">Zip/Postal Code</a></i> field must be filled in.
%elif ((error_code == "190") && (field == "cctype"))
  The <i><a href="#cc">Credit Card Type</a></i> was not selected.
  Please re-enter all credit card information.
%elif ((error_code == "190") && (field == "cctype"))
  The <i><a href="#cc">Credit Card Type</a></i> was not selected.
  Please re-enter all credit card information.
%elif ((error_code == "190") && (field == "ccnum"))
  The <i><a href="#cc">Credit Card Number</a></i> was not filled in.
  Please re-enter all credit card information.
%elif ((error_code == "1005") && (field == "ccnum"))
  The <i><a href="#cc">Credit Card Number</a></i> you entered was not valid.
  Please re-enter all credit card information.
```

```
%elif (error_code == "1006")
  The <i><a href="#cc">Credit Card Expiration Date</a></i> you entered was not valid.
  Please re-enter all credit card information.
%elif (error_code == "190")
  The <i><a href="#$(field)">$(field)</a></i> field must be filled in.
%elif (error_code == "210")
  The nickname has already been used.
%else
  An undefined error occurred.  Please contact the Site Adminstrator.  The error code is
$(error_code).
   %endif

  </B></FONT>

</TD>
</TR>

</TABLE>

<FORM method=POST action="/cgi-bin/ncommerce3/AddressUpdate">

<INPUT Type = "hidden" Name="merchant_rn" Value="$(MerchantRefNum)">
<INPUT Type = "hidden" Name="sarfnbr" Value="$(sarfnbr)">
<INPUT Type = "hidden" Name="url"
Value="/cgi-bin/ncommerce3/ExecMacro/$(STORENAME)/adrbk.d2w/report">

<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>

<TR>
<BR>

<TABLE WIDTH=500 CELLSPACING=0 BORDER=0>

<TR>
<TD COLSPAN=2><H3><FONT FACE="helvetica">Shipping Address:</FONT></H3></TD>
</TR>

<TR>
<TD COLSPAN=2>
<FONT SIZE=2>Enter the shipping address for this purchase.  An invoice will also
be sent to this address.</FONT>
</TD>
</TR>

<TR>
<TD><BR></TD>
</TR>

<TR>
<TD><A NAME="salname"><B><FONT SIZE=-1>$(l_safname)</FONT></B></TD>
<TD><A NAME="safname"><B><FONT SIZE=-1>$(l_salname)</FONT></B></TD>
</TR>

<TR>
<TD><INPUT TYPE="text" NAME="safname" VALUE="$(safname)" SIZE="25" MAXLENGTH="30"></TD>
<TD><INPUT TYPE="text" NAME="salname" VALUE="$(salname)" SIZE="25" MAXLENGTH="30"></TD>
</TR>

<TR>
<TD><A NAME="saaddr1"><A NAME="saaddr2"><B><FONT SIZE=-1>$(l_saaddr1)</FONT></B></TD>
</TR>

<TR>
<TD COLSPAN=4>
<INPUT TYPE="text" NAME="saaddr1" VALUE="$(saaddr1)" SIZE="56" MAXLENGTH="50">
</TD>
</TR>

<TR>
<TD COLSPAN=4>
<INPUT TYPE="text" NAME="saaddr2" VALUE="$(saaddr2)" SIZE="56" MAXLENGTH="50">
</TD>
</TR>

<TR>
<TD><A NAME="sacity"><B><FONT SIZE=-1>$(l_sacity)</FONT></B></TD>
<TD><A NAME="sastate"><B><FONT SIZE=-1>$(l_sastate)</FONT></B></TD>
</TR>
```

```
<TR>
<TD><INPUT TYPE="text" NAME="sacity" VALUE="$(sacity)" SIZE="25" MAXLENGTH="30"></TD>
<TD><INPUT TYPE="text" NAME="sastate" VALUE="$(sastate)" SIZE="25" MAXLENGTH="20"></TD>
</TR>

<TR>
<TD><A NAME="sazipc"><B><FONT SIZE=-1>$(l_sazipc)</FONT></B></TD>
<TD><A NAME="sacntry"><B><FONT SIZE=-1>$(l_sacntry)</FONT></B></TD>
</TR>

<TR>
<TD><INPUT TYPE="text" NAME="sazipc" VALUE="$(sazipc)" SIZE="25" MAXLENGTH="20"></TD>
<TD><INPUT TYPE="text" NAME="sacntry" VALUE="$(sacntry)" SIZE="25" MAXLENGTH="30"></TD>
</TR>

<TR>
<TD><BR></TD>
</TR>

<TR>
<TD><A NAME="saphone1"><B><FONT SIZE=-1>$(l_saphone1)</FONT></B></TD>
<TD><A NAME="saphone2"><B><FONT SIZE=-1>$(l_saphone2)</FONT></B></TD>
</TR>

<TR>
<TD><INPUT TYPE="text" NAME="saphone1" VALUE="$(saphone1)" SIZE="25" MAXLENGTH="30"></TD>
<TD><INPUT TYPE="text" NAME="saphone2" VALUE="$(saphone2)" SIZE="25" MAXLENGTH="30"></TD>
</TR>

<TR>
<TD><A NAME="saemail1"><B><FONT SIZE=-1>$(l_saemail1)</FONT><B></TD>
</TR>

<TR>
<TD COLSPAN=4><INPUT TYPE="text" NAME="saemail1" VALUE="$(saemail1)" SIZE="56"
MAXLENGTH="254"></TD>
</TR>

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

<TR>
<TD COLSPAN=2>
<H3><FONT FACE="helvetica"><A NAME="cc">Payment Information:</a></FONT></H3>
</TD>
</TR>

<TR>
<TD COLSPAN=2>
<FONT SIZE=2>Please enter your credit card information and click on the Purchase
button.</FONT>
</TD>
</TR>

<TR>
<TD>
<BR>
</TD>
</TR>



<TR>
<TD COLSPAN=2>

<TABLE WIDTH=200 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=8 >

<TR>
%if (CC_visa == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=VISA></TD>
<TD VALIGN="top"><img src="/sggifs/visa.gif"></TD>
%endif

%if (CC_master == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=MAST></TD>
<TD VALIGN="top"><img src="/sggifs/mc.gif"></TD>
```

```
%endif

%if(CC_amex == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=AMEX></TD>
<TD VALIGN="top"><img src="/sggifs/amex.gif"></TD>
%endif

%if(CC_discover == "YES")
<TD VALIGN="top"><input type=radio name=cctype value=DISC></TD>
<TD VALIGN="top"><img src="/sggifs/discover.gif"></TD>
%endif

</TR>

</TABLE>

</TD>
</TR>



<TR>
<TD COLSPAN=2>

<TABLE WIDTH=400 CELLPADDING=0 CELLSPACING=0 BORDER=0 COLS=4>

<TR><TD><BR></TD></TR>
<TR>
<TD ALIGN="left"><FONT SIZE=2>Card Number</FONT></TD>
<TD ALIGN="left"><FONT SIZE=2>Expiration Month</FONT></TD>
<TD ALIGN="left"><FONT SIZE=2>Expiration Year</FONT></TD>
</TR>
<TR>
<TD ALIGN="left" VALIGN=middle>
<INPUT TYPE=text SIZE=15 MAXLENGTH=256 NAME="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>&nbsp&nbsp
<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="2003">2004</option>
</SELECT>
</TD>

</TR>



</TABLE>
</TD>
</TR>

<TR>
```

```
<TD COLSPAN=2><BR></TD>
</TR>




</TABLE>

<TABLE WIDTH=500 CELLPADDING=0 border=0>


<TR>
<TD ALIGN="center">
<INPUT TYPE=hidden NAME=sanick VALUE=$(SESSION_ID)>
<INPUT TYPE=hidden NAME=order_rn VALUE=$(order_rn)>
<INPUT TYPE=hidden NAME=billto_rn VALUE=$(SHOPPER_REF)>
<INPUT Type = "hidden" Name="merchant_rn" Value="$(MerchantRefNum)">

<INPUT TYPE=hidden NAME="url"
VALUE="/cgi-bin/ncommerce3/OrderProcess?merchant_rn=$(MerchantRefNum)">


<input type=submit value="Resubmit Purchase">
</TD>
</TR>




</FORM>
<TR><TD COLSPAN=2><BR><HR WIDTH=550 ALIGN=left></TD></TR>
<TR><TD ALIGN="center">
<FORM action="/cgi-bin/ncommerce3/pay_wakeup">
<INPUT TYPE=hidden NAME="order_rn" value="$(order_rn)">
<INPUT TYPE=hidden NAME="merchant_rn" value="$(MerchantRefNum)">
   <input type=submit value="Pay with my Wallet for: $(order_rn)">
</FORM></TD></TR>

</TABLE>
  </BODY>
</HTML>


%}
```

A.9.12 Error Macro Bad Quantity

The Net.Data macro err_stdata.d2w is shown when the shopper enters a quantity that is not valid in the product or display current order page:

```
%include "ShopITSO/ShopITSO.inc"

%{======================================================================

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.      All Rights Reserved

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

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

%define {
 SHOWSQL="NO"
%}

%{==== Retrieves the Shopper Reference Number ====%}

%function(dtw_odbc) GET_SHOPPER_REF_NUM() {
   select shrfnbr from shopper where shlogid = '$(SESSION_ID)'
   %REPORT{
     %ROW{
       @DTW_assign(SHOPPER_REF, V_shrfnbr)
     %}
   %}
   %MESSAGE{
     default: { ERROR in GET_SHOPPER_REF_NUM %}
   %}
%}






%HTML_REPORT{
<HTML>

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

<BODY BGCOLOR="$(BodyColor1)" TEXT="$(TextCol)" LINK="$(LinkCol)" VLINK="$(VLinkCol)"
ALINK="$(ALinkCol)">

<TABLE WIDTH=500 CELLPADDING=0 CELLSPACING=0 BORDER=0>
<TR>
      <TD ALIGN="left" VALIGN="center">
  <FONT FACE="helvetica" COLOR=$(TitleTxtCol)><H3>Order Details Error</H3></FONT>
  </TD>
    </TR>
</TABLE>
@GET_SHOPPER_REF_NUM()


<TABLE>
  <TR>
<TD align=center width=85>
  <img src="/shopitso/warning.gif" valign=bottom align=left width=72 height=63>
</TD>
<TD>
  There was a problem with your submission.
  <br><P>
  %if ("$(error_code)" == "220")
You typed <B>"$(quantity)"</B> in the field <B>$(field)</B>.  A numeric value above zero is
required.
  %endif
</TD>
  </TR>
  <TR>
<TD align=center width=85>
</TD>
  <TD>
<BR>
<B><I>Click the go further button to get the order details page.</I><BR><BR>
   <I>When you come from a Product Page select the product again and type a valid
quantity.</I><BR>
   <I>When you come from the Order Details Page make the change to the quantity again.</I></b>
  </TD>
  </TR>
</TABLE>

    <BR>

<TABLE WIDTH=300>
<TR>
  <FORM ACTION="/cgi-bin/ncommerce3/OrderItemDisplay">
<INPUT TYPE="hidden" NAME="merchant_rn" VALUE="$(MerchantRefNum)">
<TD WIDTH=100 ALIGN="right"><INPUT TYPE="submit" VALUE="go further">
</TD>
  </FORM>
```

```
</TR>
</TABLE>

</BODY>

</HTML>

%}
```

A.10 ShopITSO Include File

ShopITSO.inc is built by the Store Creator and is used in all Net.Data macros as the include file. It contains define variables for the store, for example, the merchant reference number.

```
%define {
null="null"
title="Corporate (Side)"
CC_amex="NO"
CC_discover="NO"
LongStoreName="ShopITSO"
ButtonHeight="44"
ALinkCol="red"
BannerRightSpc="60"
EOF="EOF"
TextCol="black"
CC_master="YES"
NavLinkCol="#E7E7EF"
ButtonWidth="50"
BannerLeftSpc="90"
LinkCol="blue"
BannerImage="/sggifs/cr_banner_s.gif"
BannerAlign="right"
BodyImage1="/sggifs/cr_tablbkg.gif"
Button1="/sggifs/cr_butt1_s.gif"
navigation="side"
Button2="/sggifs/cr_butt2_s.gif"
Button3="/sggifs/cr_butt3_s.gif"
BannerTxtCol="#E7E7EF"
CC_phone="YES"
STORENAME="ShopITSO"
Button4="/sggifs/cr_butt4_s.gif"
BckImageNavBar="/sggifs/cr_leftbkg_s.gif"
Button5="/sggifs/cr_butt5_s.gif"
NavBarAlign="left"
gif="sggifs/cor_s.jpg"
Button6="/sggifs/cr_butt6_s.gif"
BodyColor1="#E7E7EF"
CC_visa="YES"
Button7="/sggifs/cr_butt7_s.gif"
BodyColor2="#A1A2C5"
HomeCategory="657"
MerchantRefNum="28"
Button8="/sggifs/cr_butt8_s.gif"
VLinkCol="darkblue"
BodyColor3="silver"
TitleTxtCol="black"
AddButton="/sggifs/cr_add.gif"
%}
```

A.11 AS/400 Web Server Configuration File

Here, you can find our AS/400 Web Server configuration file for our ShopITSO sample shop. It works with the HTTP instance named work.

```
Configuration: WORK

 #****************************************************
 #*** Net.Commerce/400 IBM HTTP Server Configuration **
 #****************************************************
 #
 #****************************************************
 Enable GET
 Enable HEAD
 Enable POST
 #***************************************************
 ######## IBM Net.Commerce ######## (Do not edit this section)
 Fail /logs/*
 Fail /macro/*
 Fail /nc_cache/*
 ### Protect /*.ini work.itso.ibm.com {
 Protect /*.ini work.itso.ibm.com {
 ServerId Private_Authorization
 Authtype Basic
 GetMask All@(*)
 PostMask All@(*)
 Mask All@(*)
 }
 Pass /te_html/* /Qibm/UserData/NetCommerce/instance/work/teditor/te_html/*
work.itsoroch.ibm.com
 Pass /ca_html/* /Qibm/UserData/NetCommerce/instance/work/teditor/ca_html/*
work.itsoroch.ibm.com
 Service /cgi-bin/ncommerce/* /QSYS.LIB/QNETCOMM.LIB/QNEICAPI.SRVPGM:nc_cache
 Service /cgi-bin/ncommerce3/* /QSYS.LIB/QNETCOMM.LIB/QNEICAPI.SRVPGM:nc_cache
 Service /msprotect/ncommerce/* /QSYS.LIB/QNETCOMM.LIB/QNEICAPI.SRVPGM:nc_auth
 Service /msprotect/ncommerce3/* /QSYS.LIB/QNETCOMM.LIB/QNEICAPI.SRVPGM:nc_auth
 Service /servlet/* /QSYS.LIB/QHTTPSVR.LIB/QZHJSVLT.SRVPGM:AdapterService* %%MIXED%%
 ServerInit /QSYS.LIB/QNETCOMM.LIB/QNEICAPI.SRVPGM:nc_init_cache
CACHE=OFF|CACHE_MAX_FILES=100
 ServerInit /QSYS.LIB/QHTTPSVR.LIB/QZHJSVLT.SRVPGM:AdapterInit
 /Qibm/UserData/NetCommerce/instance/work/jvm.properties
 Map /msprotect/ncommerce3/* /msprotect/ncommerce3.pgm/*
 Map /msprotect/ncommerce/* /msprotect/ncommerce3.pgm/*
 Map /cgi-bin/ncommerce3/* /cgi-bin/ncommerce3.pgm/*
 Map /cgi-bin/ncommerce/* /cgi-bin/ncommerce3.pgm/*
 Exec /msprotect/* /QSYS.LIB/QNETCOMM.LIB/*
 Exec /cgi-bin/* /QSYS.LIB/QNETCOMM.LIB/*
 DefaultFsCCSID 37
 DefaultNetCCSID 819
 Pass /storemgr/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/storemgr/*
 Pass /sitemgr/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/sitemgr/*
 Pass /ncacom/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/common/*
 Pass /ncagif/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/gif/*
 Pass /butnbars/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/butnbars/*
 Pass /ncadmin/ictmgr/*.gif /Qibm/ProdData/NetCommerce/html/MRI2924/ictimages/*.gif
 Pass /ncadmin/ictmgr/* /Qibm/proddata/netcommerce/html/ictmgr/*
 Pass /ncadmin/StoreCreator/*.class /Qibm/ProdData/NetCommerce/html/StoreCreator/*.class
 Pass /ncadmin/StoreCreator/*.jar /Qibm/ProdData/NetCommerce/html/StoreCreator/*.jar
 Pass /ncadmin/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/*
 Pass /nchelp/* /Qibm/ProdData/NetCommerce/html/MRI2924/nchelp/*
 Pass /ncerror/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncerror/*
 Pass /ncbooks/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncbooks/*
 Pass /sggifs/* /Qibm/ProdData/NetCommerce/html/MRI2924/ncadmin/StoreCreator/sggifs/*
 Pass /bus2bus2/* /Qibm/ProdData/NetCommerce/html/MRI2924/bus2bus2/*
 Pass /ca_icons/* /Qibm/ProdData/NetCommerce/html/MRI2924/ca_icons/*
 Pass /ca_widgets/* /Qibm/ProdData/NetCommerce/servlet/*
 Pass /grocery/* /Qibm/ProdData/NetCommerce/html/MRI2924/grocery/*
 Pass /teditor/* /Qibm/ProdData/NetCommerce/html/teditor/*
 AddType .js application/x-javascript binary 1.0 #Net.Commerce java
 sslmode On
 ###################################################################
 ### End of IBM Net.Commerce Entrys from NC Installation ###########
 ##############################################################
 #
 ######### New Store Section ############################################
 #
 ############################################################################
 Pass /shopitso/gifsm/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/thinkpad/sml/*
 Pass /ShopITSO/gifsm/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/thinkpad/sml/*
 Pass /shopitso/giflg/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/thinkpad/lrg/*
 Pass /ShopITSO/giflg/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/thinkpad/lrg/*
 Pass /SHOPITSO/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/*
 Pass /ShopITSO/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/*
 Pass /shopitso/* /Qibm/UserData/NetCommerce/instance/work/html/ShopITSO/*
 Pass /workonestopshop/* /Qibm/UserData/NetCommerce/instance/work/html/workOneStopShop/*
 Pass /workOneStopShop/* /Qibm/UserData/NetCommerce/instance/work/html/workOneStopShop/*
 Pass /workpersonaldelivery/*
/Qibm/UserData/NetCommerce/instance/work/html/workPersonalDelivery/*
 Pass /workPersonalDelivery/*
/Qibm/UserData/NetCommerce/instance/work/html/workPersonalDelivery/*
```

```
Pass /work1/* /Qibm/UserData/NetCommerce/instance/work/html/work1/*
Pass /work2/* /Qibm/UserData/NetCommerce/instance/work/html/work2/*
######### End of New Store Section #########
######## IBM Net.Commerce (Pass) ######## (Do not edit this section)
Pass /sample/* /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/*
Pass /* /Qibm/UserData/NetCommerce/instance/work/*
Pass / /QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/Welcome.html
######## End of IBM Net.Commerce (Pass) ########
#
#
############### DO NOT REMOVE ## OR MOVE #########
#
hostname work.itsoroch.ibm.com
BindSpecific On
CGIConvMode %%EBCDIC%%
keyfile /QIBM/userdata/icss/cert/server/server.kyr
ScriptTimeOut 5 minutes
PersistTimeout 30 seconds
ServerTerm /QSYS.LIB/QHTTPSVR.LIB/QZHJSVLT.SRVPGM:AdapterExit
#***************************************************
##### Entries for Local Cache Log #######################################
############################################################################
CacheLocalMaxBytes 2 M
CacheLocalMaxFiles 200
LiveLocalCache Off
CacheLocalFile /QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/lrg/*.gif
CacheLocalFile /QIBM/userdata/netcommerce/instance/work/html/shopitso/thinkpad/sml/*.gif
CacheLocalFile /QIBM/userdata/netcommerce/instance/work/html/shopitso/*.html
CacheLocalFile /QIBM/userdata/netcommerce/instance/work/html/shopitso/*.gif
```

A.12 INI Files

This section shows the initialization files that were used with our test set ups. You should not edit these files on your system unless you are specifically directed to in some instructions.

A.12.1 NCOMMERCE.INI

NCOMMERCE.INI is the Net.Commerce INI file for our ShopITSO sample. The PROCESSES parameter controls how many server daemons are started for an instance. The number of Net.Commerce daemons available to handle shopper requests can also affect performance when hit rates increase. The default number of daemons is two. If sufficient memory and CPU capacity is available to run additional daemons and for the database to accept additional connections, this number can be increased.

The parameter MS_HTML_MAX describes the HTML output buffer size, which you can adjust. The parameter MS_LOGLEVEL determines the amount of information written to the log file. The value of "2" writes status, error, and debug messages. When your server is ready for production, set this value to the default of "0" to reduce the overhead of generating the log information and the size of the log files.

The parameter USERTRAFFIC_LOG specifies if the user traffic log will be created in USRTRAFFIC database table. By default, the user traffic log is turned off. To turn it on, change the value from "0" to "1."

```
SERVICE_NAME_PREFIX ncmwork
EXEC /QSYS.LIB/QNETCOMM.LIB/QNESERVER.PGM
MS_HOSTNAME work.itsoroch.ibm.com
DBNAME AS01
DBINST work
DBOWNER work
DBPASS OByhZp18pJY=
PROCESSES 2
MERCHANT_KEY QunGZnDUqUDK7yW0cEnk38vOvgkAO1Ym
```

```
PDI_ENCRYPT OFF
USERTRAFFIC_LOG 0
MS_TRANS_COUNT 250
MS_HTML_MAX 1000000
MS_LOGPATH /Qibm/UserData/NetCommerce/instance/work/logs
MS_LOGLEVEL 0
DB_RETRY_LIMIT 15
DB_RETRY_INTERVAL 20
NC_DMN_CACHE 1
NC_DMN_SYNCH 1
NC_DMN_SLP_SEC 15
MACRO_PATH /Qibm/UserData/NetCommerce/instance/work/macro
HTML_PATH /Qibm/UserData/NetCommerce/instance/work/html
MS_CGIBIN_PATH /QSYS.LIB/QNETCOMM.LIB
SG_PATH /Qibm/ProdData/NetCommerce/macro/MRI2924/SmartGuide
NC_TEDITOR_PATH
/Qibm/UserData/NetCommerce/instance/work/teditor;/Qibm/UserData/NetCommerce/instance/work/te
ditor/ca_html
NC_LANG en_US
NC_WEBSERVER IBM_HTTP
NC_ENABLE_STAGING 0
CACHE_ENABLED ON
MAX_CACHED_FILES 100
CACHE_FILE_PATH /Qibm/UserData/NetCommerce/instance/work
IC_DBMS DB400
ETILL_HOSTNAME work.itsoroch.ibm.com
PAY_STATE_TIMEOUT 3600
PAYSYS_CONTROLLER 1
SECURE_NVPS password, shlpswd, shlpswdver, ccnum, ccxyear, ccxmonth, name, shlogid, sanick
IC_JDBC_DRIVER com.ibm.db2.jdbc.app.DB2Driver
IC_JDBC_NETDRIVER com.ibm.as400.access.AS400JDBCDriver
IC_JDBC_URL jdbc:db2://AS01/work
IC_JDBC_NETURL jdbc:as400://work.itsoroch.ibm.com/work
```

A.12.2 DB2WWW.INI

DB2WWW.INI is a member of the file named INI in the AS/400 QSYS library named like your instance. The member name is DB2WWW. It is the Net.Data Configuration file and has entries for the macro path to find the Net.Data macros, the include path to find include files, the EXEC path to find such DB2WWW CGI program and language environment entries as DTW_SQL and DTW_SYSTEM.

Note

This file has been reformatted for inclusion in this book. On the system, it consists of seven long records in the member DB2WWW.

```
MACRO_PATH /Qibm/UserData/NetCommerce/instance/work/macro;
           /Qibm/UserData/NetCommerce/instance/work/teditor;
           /Qibm/ProdData/NetCommerce/macro/MRI292
INCLUDE_PATH /Qibm/UserData/NetCommerce/instance/work/html;
             /Qibm/UserData/NetCommerce/instance/work/macro;
             /Qibm/UserData/NetCommerce/instance/work/teditor;
             /Qibm/ProdData/NetCommerce/macro/MRI2924
ENVIRONMENT (DTW_SQL)       /QSYS.LIB/QHTTPSVR.LIB/QTMJSQL.SRVPGM
            ( IN SHOWSQL, DB_CASE, DTW_SET_TOTAL_ROWS, DTW_EDIT_CODES, NULL_RPT_FIELD,
              OUT DTWTABLE, SQL_CODE, SQL_STATE, TOTAL_ROWS, NUM_TABLES)
ENVIRONMENT (DTW_ODBC)      /QSYS.LIB/QHTTPSVR.LIB/QTMJSQL.SRVPGM
            ( IN SHOWSQL, DB_CASE, DTW_SET_TOTAL_ROWS, DTW_EDIT_CODES, NULL_RPT_FIELD,
              OUT DTWTABLE, SQL_CODE, SQL_STATE, TOTAL_ROWS, NUM_TABLES)
ENVIRONMENT (DTW_SYSTEM)     /QSYS.LIB/QTCP.LIB/QTMHSYS.SRVPGM ()
```

A.12.3 SRVCTRL.INI

The SRVCTRL.INI server controller configuration file is used by the Server Controller daemon to determine which pools to start, which port to listen on, and how to perform logging:

```
MS_HOSTNAME work.itsoroch.ibm.com
CONTROL_DBNAME AS01
CONTROL_DBINST work
CONTROL_DBPASS OByhZp18pJY=
CONTROL_DB_RETRY_LIMIT 15
CONTROL_DB_RETRY_INTERVAL 20
MERCHANT_KEY QunGZnDUqUDK7yW0cEnk38vOvgkAO1Ym
CONTROL_ERR_TOLERANCE 1
CONTROL_SERVICE ncmworkctrl
CONTROL_POOL_CONFIG /Qibm/UserData/NetCommerce/instance/work/ncommerce
MS_LOGPATH /Qibm/UserData/NetCommerce/instance/work/logs
```

A.12.4 PAY_BACK.INI

The PAY_BACK pool configuration file controls how background payment requests are processed:

```
SERVICE_NAME_PREFIX ncmwork_pb
EXEC /QSYS.LIB/QNETCOMM.LIB/QNEBACKSVR.PGM
DBNAME AS01
DBINST work
DBPASS OByhZp18pJY=
PROCESSES 1
MERCHANT_KEY QunGZnDUqUDK7yW0cEnk38vOvgkAO1Ym
MS_HOSTNAME work.itsoroch.ibm.com
DB_RETRY_LIMIT 15
DB_RETRY_INTERVAL 20
ETILL_HOSTNAME work.itsoroch.ibm.com
BACKGROUND_CYCLE_TIME 30
PAY_STATE_TIMEOUT 3600
BACKGROUND_INTER_JOB_TIME 30
MS_LOGPATH /Qibm/UserData/NetCommerce/instance/work/logs
MS_LOGLEVEL 0
PAYSYS_CONTROLLER 0
```

A.12.5 PAY_ETILL.INI

The PAY_ETILL pool configuration file controls the startup and operation of the eTill server:

```
SERVICE_NAME_PREFIX ncmwork_pe
EXEC /QSYS.LIB/QNETCOMM.LIB/QNESERVER.PGM
DBNAME AS01
DBINST work
DBPASS OByhZp18pJY=
PROCESSES 1
MERCHANT_KEY QunGZnDUqUDK7yW0cEnk38vOvgkAO1Ym
MS_HOSTNAME work.itsoroch.ibm.com
DB_RETRY_LIMIT 15
DB_RETRY_INTERVAL 20
USEREXIT_HOSTNAME work.itsoroch.ibm.com
USEREXIT_SERVICE_NAME_PREFIX ncmwork
ETILL_HOSTNAME work.itsoroch.ibm.com
ETILL_EXEC QSYS/STRPYMSVR
MS_LOGPATH /Qibm/UserData/NetCommerce/instance/work/logs
MS_LOGLEVEL 0
PAYSYS_CONTROLLER 0
```

A.12.6 INSTANCE.INI

Here is the INI file for our instance named work. Do not edit this file:

```
NC_ENABLE_SEP_WEB_SERV 1
NC_USE_Payment_SVR 0
LANGUAGE 0
CCSID 37
LANGID ENU
LOCALE /QSYS.LIB/EN_US.LOCALE
FSCCSID 37
NETCCSID 819
DBPOPULATE 0
ENABLE_SIMPLIFY_CACHE 0
```

Appendix B. Performance

This appendix briefly describes some AS/400-specific tips and techniques that may help improve your site performance.

B.1 Using the DNSLookup Directive

Use the DNSLookup directive in the HTTP configuration file to specify whether you want the server to look up the host name of requesting clients. When you enable Access and Error logging, the log contains either the IP address (DNSLookup OFF) or the actual host name (DNSLookup ON) of the requesting clients. Setting DNSLookup to OFF conserves network and AS/400 resources by eliminating the host name lookup for each log entry. In many cases, the client IP address is sufficient to track server activity, and an address lookup yields a proxy name, not the name of the remote system.

To update the HTTP server configuration file, use the Work with HTTP Configuration (`WRKHTTPCFG`) command or the ADMIN HTTP server to specify the following entry in the HTTP instance configuration file:

- To specify that clients be identified in log files by host name, enter:

  ```
  DNSLookup ON
  ```

  This can reduce the HTTP server performance.
- To specify that clients be identified in log files by the IP address, enter:

  ```
  DNSLookup OFF
  ```

B.2 Tuning SQL Requests

When writing new macros for your site implementation, use SQL to select rows from the Net.Commerce database. It is important that you write the SQL statements as efficiently as possible. The following resources can help you with SQL performance tuning:

- For a detailed description of the DB2/400 Optimizer, as well as some tips and techniques to improve query performance, refer to Chapter 22 in *DB2 for AS/400 SQL Programming Version 4*, SC41-5611.
- You can find useful advise about query optimization and performance problem determination on the Web at: `http://www.as400.ibm.com/developer/client/performance/csperdg5.html`
- For database technical papers and performance improvements tips, go to the site at: `http://www.as400.ibm.com/db2/db2tch_m.htm`
- For specific net.data related performance issues, go to the site at: `http://www.as400.ibm.com/netdata`

B.3 Increasing the Max Active Value of the Memory Pool

During our testing, we found that when increasing the Max Active value in the *BASE pool (where Net.Commerce runs) to a high number (in our case, 1500), Net.Commerce ran better. This is based on very unscientific measurements. If you change this setting, monitor the system performance to ensure that the change does not have an adverse effect on the system. One reason this may help is that each thread requires an activity level. Some of the functions used by Net.Commerce use threads to perform their functions. In some cases, the system counts a thread as "active" even when it is not doing productive work. This can cause a large number of the activity levels to be taken up without producing real work.

B.4 Adjusting the QNETCOMM Jobs Priority

If your Net.Commerce server is serving other applications, you may want to allocate more resources to the Net.Commerce server jobs. You can increase the Net.Commerce servers priority by using the `CHGCLS` command. Type `CHGCLS` and press **F4**. Complete the command parameters as described in Figure 460.

```
                                 Change Class (CHGCLS)

 Type choices, press Enter.

 Class  . . . . . . . . . . . . . > QNETCOMM      Name
   Library  . . . . . . . . . . . >   QNETCOMM    Name, *LIBL, *CURLIB
 Run priority . . . . . . . . . .   20            1-99, *SAME
 Time slice . . . . . . . . . . .   2000          Milliseconds, *SAME
 Eligible for purge . . . . . . .   *YES          *SAME, *YES, *NO
 Default wait time  . . . . . . .   30            Seconds, *SAME, *NOMAX
 Maximum CPU time . . . . . . . .   *NOMAX        Milliseconds, *SAME, *NOMAX
 Maximum temporary storage  . . .   *NOMAX        Kilobytes, *SAME, *NOMAX
 Maximum threads  . . . . . . . .   *NOMAX        1-32767, *SAME, *NOMAX
 Text 'description' . . . . . . .   *BLANK
```

Figure 460. Changing the Net.Commerce Server Run Priority

The shipped value of the Net.Commerce server jobs priority is 25. In the example, we changed it to 20.

Note

If you increase the Net.Commerce server jobs priority, other jobs on the system may decrease their performance.

B.5 Loading Net.Commerce Tables to Main Memory

The AS/400 system supports moving objects to main memory by using the SETOBJACC command. In some cases, you may find it beneficial to load a Net.Commerce product-related table, such as products and prodprcs, to main memory at the beginning of the day. This command can change the speed of accessing the product information and improve the site performance.

The CL program shown in Figure 461 on page 527 is an example of loading Net.Commerce products data into main memory.

```
/********************************************************/
/* This program will load data from Net.Commerce product   */
/* related tables to the main memory.                      */
/* =====================================================*/
/* This is only an example. You must check if this approach*/
/* will improve performance in your specific Net.Commerce  */
/* site.                                                   */
/* =====================================================*/
/* Author: Shahar mor                                      */
/* Provided AS IS                                          */
/********************************************************/
             PGM

             DCL        VAR(&msgid)      TYPE(*CHAR) LEN(7)
             DCL        VAR(&msgdta)     TYPE(*CHAR) LEN(80)

/* Create the subsystem if not active yet */
             CHKOBJ     OBJ(QGPL/NETCMEM) OBJTYPE(*SBSD)
             MONMSG     MSGID(CPF0000) EXEC(DO)
             CRTSBSD    SBSD(QGPL/NETCMEM) POOLS((1 3000 1)) +
                          TEXT('Load Net.Commerce objects to memory')
             ENDDO

/* Start the subsystem to allocate main storage */
             STRSBS     SBSD(QGPL/NETCMEM)
             MONMSG     MSGID(CPF1010) /* Already active */


/* Load objects. */
             CLRPOOL    POOL(NETCMEM 1)
             SETOBJACC  OBJ(WORK/PRODUCT ) OBJTYPE(*FILE) +
                          POOL(NETCMEM 1)
             SETOBJACC  OBJ(WORK/PRODPRCS) OBJTYPE(*FILE) +
                          POOL(NETCMEM 1)
             SETOBJACC  OBJ(WORK/PRODATR ) OBJTYPE(*FILE) +
                          POOL(NETCMEM 1)
             SETOBJACC  OBJ(WORK/PRODSGP) OBJTYPE(*FILE) +
                          POOL(NETCMEM 1)
             RCVMSG     MSGTYPE(*LAST) MSGDTA(&msgdta) MSGID(&msgid)
             IF         COND(&msgid = 'CPC1140') THEN(DO)
                 IF         COND(%BIN(&msgdta 42 4) < %BIN(&msgdta 38 +
                          4)) THEN(DO)
                    SNDUSRMSG  MSGID(CPF9897) MSGF(QCPFMSG) MSGDTA('====== +
                             Note : Net.Commerce product tables were +
                             not completly loaded !!') MSGTYPE(*INFO) +
                             TOMSGQ(*SYSOPR)
                 ENDDO
             ENDDO

             ENDPGM
```

Figure 461. Example to SETOBJACC Usage

For more information about the SETOBJACC command, see *Work Management Version 4*, SC41-5306.

Note

Make sure you have enough main memory before implementing the SETOBJACC command. Also note that using the SETOBJACC command does not always improve performance.

If you do not have enough memory for loading the Net.Commerce tables data, you can gain performance by loading only the tables access path.

B.6 Improving the IBM Client Access ODBC Driver Performance

In Chapter 18, “Generating Net.Commerce Reports” on page 403, we describe the possibility of generating Net.Commerce reports using an ODBC connection to the AS/400 database. If your Net.Commerce schema has many rows, you may find the ODBC connection to be slow. In that case, you may consider using the IBM Client Access Driver compression.

Data compression can improve performance of the IBM Client Access ODBC driver. This data compression enhancement (delivered through the V4R2 PTF, SF49349, and the V4R3 PTF, SF51501) reduces the amount of data that has to be sent when returning ODBC results. When less data is transferred, the performance is faster.

For compatibility reasons, the data compression is not the default behavior. The result data must be compressed by requesting data compression at the connection or statement level. The graphical interface for configuring data sources does not include support for the data compression option. Therefore, the entry must be added manually.

For more techniques and tips for improving ODBC performance, see the IBM Partners in Development Web site at:

`http://www.as400.ibm.com/developer/client/odbc.html`

Appendix C. Problems and Solutions

This appendix includes a few tips that we found for both Net.Commerce and Net.Data.

C.1 Net.Commerce

This section offers some additional tips that we found relating to Net.Commerce during our testing.

C.1.1 Net.Commerce Online Documentation

Net.Commerce product installation places the help and online documentation on the AS/400 server. To read the product online help, use your browser and map a PC network drive to your AS/400 server. For example, use the following procedure to read the online help from Netscape Navigator 4.5:

1. Double click on the Netscape icon to launch Netscape.
2. Choose **File** —> **Open page**.
3. Click on the **Choose file** button.
4. Locate your server and point to the directory: /QIBM/ProdData/NetCommerce/html/MRI2924/nchelp
5. Double-click on the file **index.htm**.

   The online help now appears on the browser.

The Net.Commerce manuals are located in the directory /QIBM/ProdData/NetCommerce/html/MRI2924/ncbooks. The manuals are in a PDF format so you need the Adobe Acrobat Reader to view the manuals. You can download a free version of Adobe Acrobat Reader from the Web at:
`http://www.adobe.com`

C.1.2 The ExecMacro Command

Net.Commerce provides the ExecMacro command to freely call any Net.Data macro. However, exercise caution with this command because it allows totally unprotected access to the macro. Access control, security checks, or specific business logic are not executed with this command.

In addition, the ExecMarco command does not use the Net.Commerce cache when displaying the product and category pages. The ExecMacro command is best used during prototyping because it offers a quick way to implement functionality that will later be converted to commands or OFs.

C.2 Net.Data and Net.Commerce

Net.Data is used as the main display engine for the Net.Commerce system. As a result, Net.Data should not be used to implement business logic.

Implementing business logic is best left to commands and OFs, instead of Net.Data. Although Net.Data supports SQL statements such as INSERT, UPDATE, and DELETE, we strongly recommend that these statements never be used in any macro in a Net.Commerce system. Doing so directly contradicts the general programming model of Net.Commerce. Commands and OFs are the recommended engine for database inserts or updates. Net.Data is used as a display tool only.

The Net.Data product that is used with Net.Commerce is the same Net.Data version used by the rest of the system. Net.Data has been enhanced to support Net.Commerce. Due to some of the security functions added in Net.Commerce, some Net.Data functions may not be used when accessed through Net.Commerce.

C.2.1 Error Handling

SQL functions contain a message block where the results of non-zero SQL return codes are processed. The most common use for this message block is to process the SQL code 100, where no rows were returned from the query. However, the message block should always have a default section where all error codes that are not explicitly caught are processed. If default error sections in message blocks are omitted, instead of handling an error intelligently, the macro ignores the error.

In all situations, all possible error codes that can occur during normal operation should be dealt with appropriately (for example, setting a default value if one was not found in the database). In addition, a default section should be used to process all other errors in an intelligent manner (such as displaying an error and stopping the execution of the macro).

C.2.2 Performance Considerations

Under high-volume and stressed environments, performance is key. A good way to improve performance without affecting functionality is to remove all unnecessary white space. When you write Net.Data macros, every character or white space in the file is sent to the browser. White space wastes bandwidth.

Also, function calls and other control syntax, such as %if blocks in Net.Data, produce a new line character in the HTML output. If these elements are in a %ROW section, and if numerous rows are returned, this can create excessive white space.

Optimizing those %ROW sections by improving the SQL and removing all unnecessary white space and function calls dramatically improves performance. The side effect of this is that the macro becomes less readable.

This code fragment demonstrates a good use of white space:

```
%REPORT{
%ROW{
@DTW_assign(TAX_RATES, V_mttaxrate6)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate6,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate4,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate3,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate2,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate1,TAX_RATES,TAX_RATES)
%}
%}
```

Although this code fragment produces the same output in the browser, several characters of white space are transmitted where the extra carriage returns are inserted, which impacts performance:

```
%REPORT{



%ROW{


@DTW_assign(TAX_RATES, V_mttaxrate6)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate6,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate4,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate3,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate2,TAX_RATES,TAX_RATES)
@DTW_concat(";",TAX_RATES,TAX_RATES)
@DTW_concat(V_mttaxrate1,TAX_RATES,TAX_RATES)



%}
%}
```

Appendix D. Special Notices

This publication is intended to help people plan and implement Net.Commerce sites on the AS/400 system. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM Net.Commerce for AS/400 or IBM Payment Server V1.2 for AS/400. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Net.Commerce for AS/400 or IBM Payment Server V1.2 for AS/400 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

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 them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications, see "How to Get ITSO Redbooks" on page 537.

- *AS/400 Internet Security: IBM Firewall for AS/400*, SG24-2162
- *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*, SG24-4978
- *DB2/400 Advanced Database Functions*, SG24-4249
- *Lotus Domino for AS/400 - Installation, Customization, and Administration*, SG24-5181
- *IBM Firewall for AS/400 V4R3: VPN and NAT Support*, SG24-5376
- *Building e-Commerce Solutions with Net.Commerce: A Project Guidebook,* SG24-5417

E.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at `http://www.redbooks.ibm.com/` for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr Format)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037

E.3 Other Publications

These publications are also relevant as further information sources:

- *Basic System Operation, Administration, and Problem Handling Version 4,* SC41-5206
- *Work Management Version 4*, SC41-5306
- *Tips and Tools for Securing Your AS/400 Version 4*, SC41-5300
- Communications Management Version 4, SC41-5406
- *TCP/IP Configuration and Reference*, SC41-5420
- *Getting Started with IBM Firewall for AS/400*, SC41-5424

- *DB2 for AS/400 SQL Programming Version 4*, SC41-5611
- *TCP/IP tutorial Technical Overview*, GG24-3376
- *Net.Commerce for AS/400 Installing and Getting Started Guide,* GC09-2864
- *IBM HTTP Server for AS/400 Webmaster's Guide*, GC41-5434
- *Installing and Managing Domino for AS/400*, Part No. 12999
- *Extending the Domino System*, Part No. 12953s
- Leland, David. September 1998. RPG Utility puts QtmmSEndMail API to Work. *NEWS/400 Magazine* (`http://www.news400.com/code/newscode/`).

E.4 Other Resources

These Web sites offer relevant information sources. Here, you will find such documents as *Commands, Tasks, Overridable Functions, and the E-commerce Framework*:

- The Net.Commerce Web site at:
  `http://www.software.ibm.com/commerce/net.commerce/`

  Select the product, version, platform, language, and resource type. In addition to looking at the AS/400 information, look at the information and utilities provided for other platforms such as AIX or NT. You will find that, in some cases, these can be used with your AS/400 system.
- The Web site `http://www.as400.ibm.com/misc/map.htm` contains links to various AS/400 products and their PTFs
- The Web site `http://www.as400.ibm.com/http` contains a link to a list of PTFs available for the IBM HTTP Server for AS/400 and IBM WebSphere Application Server 1.1

The Net.Commerce documents directory of your AS/400 system is another place that you may find useful information.

This directory is available on your AS/400 system after you install the Net.Commerce product. The directory path is:
/QIBM/ProdData/NetCommerce/html/MRI2924/ncbooks

Refer to C.1.1, "Net.Commerce Online Documentation" on page 529, for details about accessing the documentation.

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 Redbook Evaluation

Net.Commerce V3.2 for AS/400: A Case Study for Doing Business in the New Millennium
SG24-5198-00

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!)

SG24-5198-00

Printed in the U.S.A.

Net.Commerce V3.2 for AS/400: A Case Study for Doing Business in the New Millennium

SG24-5198-00