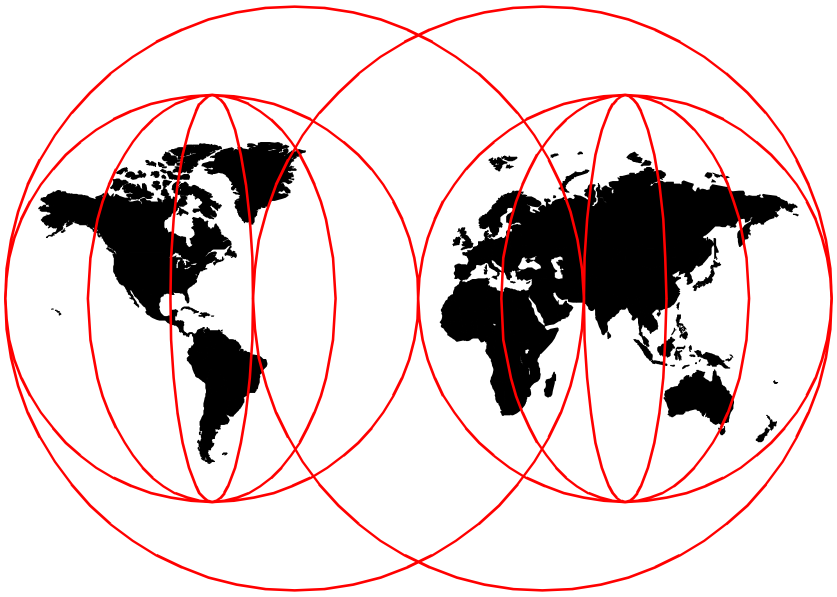
IBM

Net.Commerce: Develop on NT or AIX, Deploy on S/390

Don Bagwell, Mike Foster, Heath LaCoss



International Technical Support Organization

www.redbooks.ibm.com

SG24-5516-00

SG24-5516-00

International Technical Support Organization

Net.Commerce: Develop on NT or AIX, Deploy on S/390

March 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix A, “Special notices” on page 61.

First Edition (March 2000)

This edition applies to Version 3.1.2 of Net.Commerce for OS/390, Program Number 5697-D32 for use with the OS/390 Version 2 Release 6.

This document created or updated on March 22, 2000.

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

Contents

Preface

This redbook will help in understanding how to develop OS/390 Net.commerce solutions using Windows NT or AIX platforms. The approach presented in this redbook allows OS/390 Net.Commerce application development activity to be performed on either the Windows NT or AIX platform and then be ported to an OS/390 server for final testing and deployment.

The approach outlined in this redbook allows for full advantage of the features and capabilities of both the development and deployment platforms. By allowing designers of e-business solutions to work on the Windows NT or AIX platform enables them to maximize their productivity through the use of workstation tools along with the flexibility that these independent systems afford the developer. When testing and debugging is completed on the workstation, the application is then ported to OS/390 for final testing and placed into production. By placing an application into production on the S/390 server, allows full advantage of the reliability, scalability, and security strengths of the S/390 environment along with access to all key business applications and data of the S/390 host system.

The experience of a customer's implementation of the concepts presented in this redbook is included along with lessons learned during the project. Using the insight gained in this implementation can increase the success of projects while improving application development activity.

A working knowledge of the various environments supported by Net.Commerce is assumed. These include the operating systems referenced in this redbook: Windows NT, AIX, and OS/390, as well as the DB2 relational database management system and the functions of the Internet and Web servers.

The team that wrote this redbook

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

Don Bagwell is a Senior IT Specialist in the Washington Systems Center, located in Gaithersburg, Maryland. He has 16 years of experience with IBM in subject areas as varied as logistics management, software testing, networking sales and support, RS/6000 sales, and OS/390 technical support. For the past three years, he has focused on IBMs e-commerce suite running on OS/390. He focuses on technical education and technical marketing and has worked with several customers on their Net.Commerce/390 implementations.

Mike Foster is a senior programmer at the ITSO, Austin Center and holds a Bachelor of Science degree in Electrical Engineering from the University of Kansas. He writes extensively and teaches classes world-wide on a variety of host and PC based products. Before joining the ITSO in 1995, he held both management and technical positions in IBM marketing and development divisions supporting MVS, CICS, and DB2 products world-wide for over 25 years.

Heath LaCoss is a S/390 IT Specialist for e-commerce. He holds a Bachelor of Science degree in Electrical Engineering from the University of Texas at Austin where he served as Chair of the Institute of Electronics and Electrical Engineers for two years. Before joining S/390, he served as an intern in both the Gigahertz Processor and the RS/6000 Configurator Development groups.

Thanks to the following people who wrote the redbook, *Migrating Net.Commerce Applications to OS/390*, SG24-5438, that provided detail technical information for this redbook:

Erich Amrehn
Certified Senior IT Specialist at the International Technical Support Organization, Austin Center

Daniel Bourque
IBM Santa Teresa Software Laboratory located in San Jose, California

Nick Carlin
IBM Beta Program Manager for Net.Commerce/390 for Europe and Middle East, Hursley Park in the United Kingdom

Vinod V. Kumar
IBM Technical Manager in Learning Services for IBM Global Services India

David Hauser
IBM Santa Teresa Software Laboratory located in San Jose, California

Comments welcome

Your comments are important to us!

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

- Fax the evaluation form found in “IBM Redbooks review” on page 85 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com/`
- Send your comments in an Internet note to `redbook@us.ibm.com`

Chapter 1. Overview

This book will explore how to develop an OS/390 Net.Commerce solution on a Windows NT or AIX platform to be deployed on an OS/390 system for testing and, ultimately, placement into production. The reason this is being written is because many people desire the flexibility of the Windows NT or AIX environment for development. The OS/390 platform is the intended target for the production system because of the strengths of S/390 in reliability, scalability, and security.

This redbook is *not* a complete “how to” on developing a Net.Commerce application. Rather, this document will cover the issues needed to be aware of to avoid problems when employing dual environments.

It is also assumed that the reader has a fair knowledge of OS/390 and UNIX concepts. Examples are offered of certain commands that will assist in understanding, but this redbook will not explain, for example, the concept of “mount points” and “file system”.

1.1 Why develop on another platform?

The OS/390 has, for decades now, been the choice for companies when it comes to hosting the applications that run their business. This is because the OS/390 environment has a well developed set of tools and facilities to maintain and manage the system. True 24 x 7 availability is possible with OS/390.

Because OS/390 systems are typically very tightly controlled (because of the importance of the applications running on them), development of new solutions on an OS/390 box is usually tightly controlled as well. Contrast this with an Windows NT or AIX box, where the developer is free to make modifications and reboot pretty much at will.

When it comes to developing Net.Commerce solutions, most of the skill base exists for the Windows NT or AIX platforms and not OS/390. This is due, in large part, to the fact that there are more NT/AIX Net.Commerce opportunities than there are OS/390 opportunities. The skill base is larger, and it is easier to keep skills sharp. So, to develop a Net.Commerce solution for OS/390, skilled people most likely need to be employed on Net.Commerce for the Windows NT or AIX platform.

If you were to take a team of Windows NT or AIX people and put them in front of a 3270 terminal, you would probably be met with a look of “you’re kidding, right?” And, to ask them to learn S/390 would imply too much time spent doing that and not enough time developing the application itself. Conversely, if you were to take experienced, MVS programmers and tell them to develop a Net.Commerce application, they’d probably say “you’re kidding, right?” They too would require too much time to learn all the nuances of Net.Commerce.

So you have a dilemma, right?

Not necessarily. By taking the NT/AIX Net.Commerce people and providing them with an environment with which they are familiar, you facilitate the development of the application. And by insulating them from much of the OS/390 structure, you get the advantages of the OS/390 platform without having to teach them all about MVS.

By developing on an Windows NT or AIX platform, you provide good flexibility and a familiar environment. Because the Net.Commerce product is so comparable between platforms (as long as the version numbers are the same), you have an excellent opportunity to have the best of both worlds.

1.2 Team structure

Net.Commerce development teams can take many different forms. They can be composed of OS/390 skill sets only, Windows NT or AIX skill sets only, or a mix of the two. Obviously, as discussed in 1.1, “Why develop on another platform?” on page 1, the first alternative is highly unlikely because the skills just aren’t that common. A team composed of Windows NT or AIX skills only would not be able to take advantage of some of the capabilities of OS/390 itself simply because the Windows NT or AIX people wouldn’t have the skill to exploit them. A blended team, therefore, yields the best result.

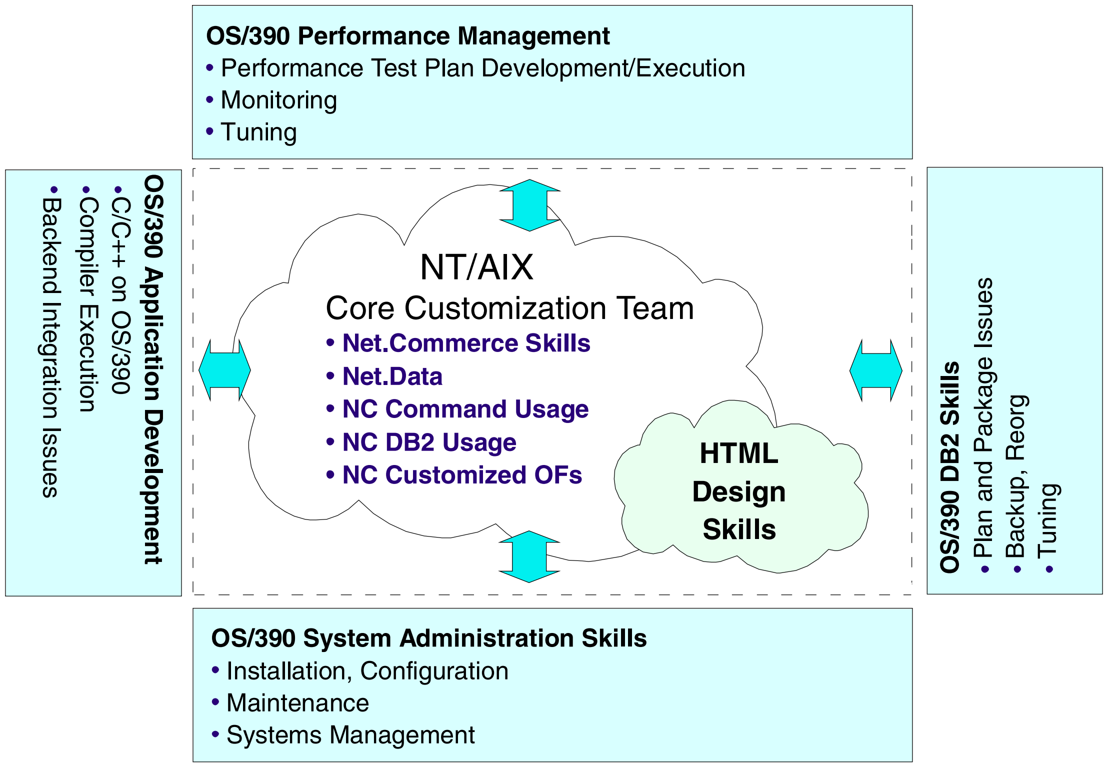


Figure 1. The role of OS/390 skills

Figure 1 illustrates how Windows NT, AIX and OS/390 skills would work together. The Windows NT or AIX developers would not need to know OS/390 specifically, but they should be aware that OS/390 is the platform on which the final solution should run. At the same time, the OS/390 team would not need to know the intricacies of Net.Commerce. They would need to know the concepts of Net.Commerce and what the Windows NT or AIX team is attempting to accomplish.

The OS/390 disciplines that should be used are shown in the chart. This can be the same person (if they possess all these skills), or a combination of people:

OS/390 Systems Administration -- Needed to install, configure, and administer the 390 environment. This is something the Windows NT or AIX people would not be able to do. These OS/390 people "shield" the Windows NT or AIX people from the administration of the OS/390 box.

OS/390 Application Development -- Needed to help the Windows NT or AIX people with compiling their code on OS/390 as well as any issues with integration with backend OS/390 systems.

OS/390 DB2 Skills -- Windows NT or AIX people may be familiar with DB2 on Windows NT or AIX, but they may stumble when working on the OS/390 system. A good OS/390 database administrator (DBA) is helpful. Often, this skill will come from the customer, as their DBAs are usually the most familiar with their DB2 system.

OS/390 Performance Management -- Stress testing the Net.Commerce solution is one of the last things you do, and it requires an extensive knowledge of overall S/390 tuning to accomplish the test. To the Windows NT or AIX people, this area is a complete mystery. To many OS/390 people, this is a complete mystery. To the extent you can get good skills in OS/390 performance management, the better your project will be.

1.3 Concurrency of development on AIX/NT and OS/390

It is important when developing Net.Commerce sites to keep the development cycles on the Windows NT or AIX machines closely in line with that of the S/390. In other words, updates to the S/390 should be made regularly, and the project should *not* be looked on as a migration effort. Much of the design and testing of individual pieces of the overall project can be performed on the Windows NT or AIX box. To what extent you can do this depends quite a bit on how much integration with backend systems is involved. But, a good deal of development is usually possible. Once development is done on the Windows NT or AIX box, it can be moved up to S/390 where the solution should be tested again.

As the project progresses forward, more and more of the work will be done on OS/390. However, if a Net.Data macro needs modification, that work will be done on the Windows NT or AIX box first, validated, then shipped back up to OS/390. The same holds true for C++ code: If a command or OF needs modification, it will be updated using the source on Windows NT or AIX, compiled to make sure that runs clean, then the source will be shipped up to OS/390 and compiled for execution on that environment. This gives a good deal of freedom to experiment before integration on S/390.

1.4 Compatibility of Net.Commerce on different platforms

If you choose to use a dual development environment, such as we're describing here, it's important to insure you have the same level of

Net.Commerce on all the platforms. Starting with Net.Commerce Version 3.1.2, the product is, for the most part, functionally equivalent across all the platforms. What you would *not* want to do, for example, is have Net.Commerce V3.1.2 on the Windows NT or AIX platform and Net.Commerce V3.1 on the OS/390.

Note

Keep the level of Net.Commerce the same on all platforms, and don’t use any version of Net.Commerce below Version 3.1.2.

1.5 AIX or NT for development

Both NT and AIX are acceptable platforms for development. If you have a choice, and you really don’t have a preference, then choose AIX. The AIX environment closely mirrors the UNIX environment of OS/390. Under NT, for example, you must change back slashes to forward-slashes in any Net.Data macro or HTML code before moving the piece onto S/390. On the other hand, the actual hardware to run AIX is more expensive than the hardware needed for Windows NT. In either case, development can occur rapidly and relatively easily before the project is moved to S/390.

Chapter 2. Establishment of environments

Because Net.Commerce V3.1.2 is more or less equal in function on Windows NT or AIX as well as OS/390, you may remotely develop Net.Commerce on Windows NT or AIX and deploy on OS/390. What is important is that the remote systems must be capable of connecting to the OS/390 systems. For example, the developers working on the Windows NT or AIX machines must be able to Telnet or FTP into the OS/390. The systems must also be adequately sized so that development is not too slow. Compiling code on an old 200 MHz system with 64 MB of memory will naturally take more time than on one of the new 500 MHz machines with 2 GB of memory.

The version of Net.Commerce used on both the remote platform and the S/390 should be the same. For instance, if Net.Commerce v3.1.2 is used on the S/390, the same version should be used for development on Windows NT or AIX. The issues involved with developing on one level of Net.Commerce and implementing on another are complex and should be avoided.

The following sections provide suggestions for configurations to use for development.

2.1 NT

Development of Net.Commerce solutions can be developed on a variety of Windows NT configurations. The main consideration for developing on Windows NT is that compatible versions of Net.Commerce, Net.Data, and DB2 be used, and a connection to the S/390 system be capable of being established.

2.1.1 Software

To support the development of your Net.Commerce applications on Windows NT, you need to establish a platform with the following software installed:

- Windows NT Server V4.0 with Service Pack 3.
- Web Browser that supports Secure Sockets Layer (SSL), Cookies, Java 1.1.5, JavaScript, tables and frames (for example, Netscape 4.0.6 or later).
- Lotus Go Web Server V4.6.2.51 or Lotus Domino V4.6.
- IBM DB2 UDB V5 or Oracle 8.0.3.
- JDK V.1.1.7.
- The compiler for Net.Commerce 3.1.2 is Microsoft Visual C++ V4.2.

- Net.Commerce Version 3.1.2.

2.1.2 Hardware

The following hardware will support the development of your Net.Commerce application using the Windows NT platform:

- IBM PC or compatible.
- Pentium 166 MHz or faster.
- 64 MB RAM (96 MB recommended).
- 400 MB free disk space (minimum).
- LAN adapter to support TCP/IP protocol.

2.2 AIX

Development of Net.Commerce solutions can also be developed on a variety of AIX configurations. The main consideration for developing on AIX is that compatible versions of Net.Commerce, Net.Data, and DB2 be used, and a connection to the S/390 system be capable of being established

2.2.1 Software

To support the development of your Net.Commerce applications on AIX, you need to establish a platform with the following software installed:

- AIX V4.1.5, V4.2.1 or later.
- Web Browser that supports: SSL, Cookies, Java 1.1.5, JavaScript, tables and frames (for example, Netscape 4.0.6 or later).
- Lotus Go Web Server V4.6.2.51, Lotus Domino V4.6, or Netscape Enterprise Server v3.0.
- IBM DB2 UDB V5.
- Net.Commerce Version 3.1.2.

2.2.2 Hardware

The following hardware will support the development of your Net.Commerce application using the AIX platform:

- IBM RS/6000 model C20 or higher.
- 120 MHz PowerPC 604 processor.
- 128 MB RAM (minimum).

- 500 MB free disk space (minimum).
- LAN adapter to support TCP/IP protocol.

2.3 OS/390

The S/390 configuration used to develop the solution should be considered separately from the configuration used for the production environment. For development, any hardware configuration capable of supporting the prerequisite software will do. In most customer environments, a development system will exist, typically an LPAR, and it will usually suffice. The time it takes to compile code may not be to your liking, but it will work.

The key is being able to install and operate the required software.

2.3.1 Software

To support the testing of your Net.Commerce applications, you will need to establish an OS/390 environment with the following software installed:

- OS/390 V2.6 or higher because of the TCP/IP performance offered in this version and higher.
- TCP/IP with FTP and Telnet established.
- IBM Domino Go 4.6.1 or IBM HTTP Server 5.1.
- DB2 V5.1 or higher.
- Net.Commerce 3.1.2.
- JDK 1.1.8 (if you plan to use Java).
- C/C++ (if you plan to customize commands or overrideable functions).

2.3.2 Hardware

As stated earlier, there is no minimum or ideal level of hardware required for Net.Commerce development other than it must be capable of running the prerequisite software. Most customers will have a development system already in place, and you may simply use what is already in place.

2.4 OS/390 Net.Commerce custom directories

When developing your Net.Commerce solution, you will want to keep custom code (HTML files, Net.Data macros, and custom overrideable functions) in their own directories. You could put them into the default directory structure of

Net.Commerce, but you stand the chance of losing some of your custom code if a PTF comes out that overlays a directory.

Providing custom directories involves two steps: Creating the directories in the hierarchical file system (HFS) and then telling the Net.Commerce system to look in those new directories when the file is requested. We'll assume you know how to create directories in the HFS. In this section, we'll discuss how to tell Net.Commerce about the location of these new directories.

2.4.1 Custom HTML directory

HTML files are not served by the Net.Commerce application. Rather, they are served to the browser by the HTTP server. As such, it is the HTTP server that must be told of any new custom directory created to hold the solution's custom HTML files.

This is done with `Pass` directives in the httpd.conf file. The `Pass` directives tell the HTTP server how to map a URL to the HFS directory in which the HTML file resides.

Assume you created a custom directory for HTML files at this location:

```
/usr/lpp/NetCommerce/html/en_US/<instance>/custom_html
```

The update in the httpd.conf file would look like this:

```
Pass /custom_html/* /usr/lpp/NetCommerce/html/en_US/<instance>/custom_html
```

Where `<instance>` is the name of the Net.Commerce instance. When the HTTP Server receives a URL of:

```
http://<host name>/custom_html/welcome.html
```

It will map the `custom_html` portion of that URL to the full HFS directory location and retrieve the file, welcome.html, from that directory.

You must stop and start the HTTP Server to enable any changes made to the httpd.conf file.

2.4.2 Custom Net.Data macro directory

Net.Data macros are invoked based on Net.Commerce commands received from the browser. To indicate the creation of a custom HFS directory to Net.Data (which is provided wrapped under the covers of Net.Commerce), you would update the db2www.ini file. The db2www.ini file contains two directives, `MACRO_PATH` and `INCLUDE_PATH`, which tells Net.Data where to look for macros or where to include files. Net.Data will search through the directories provided on the directives until it finds the files referenced.

The db2www.ini file is located in the Net.Commerce document root, which is:

```
/usr/lpp/NetCommerce/html/en_US/<instance>
```

where `<instance>` is the name of the Net.Commerce instance.

Assume you created a custom directory for Net.Data macros at this location:

```
/usr/lpp/NetCommerce/macro/en_US/<instance>/custom_macro
```

The update in the db2www.ini file would look like this:

```
MACRO_PATH /usr/lpp/NetCommerce/macro/en_US/<instance>/custom_macro; ...
```

And the update to `INCLUDE_PATH` would be the same format.

You must stop and start Net.Commerce to pick up the changes to the db2www.ini file.

2.4.3 Custom Net.Commerce command and OF directory

Net.Commerce commands and overrideable functions are invoked when a command is received in the URL. The process by which Net.Commerce determines which “shared object” (*.so file) to invoke is to look in the OFS table in the database and find the file name for the overrideable function and call it accordingly. Where it looks for that file is determined by the `LIBPATH` directive specified in the ncommerce.envvars file.

The ncommerce.envvars file is maintained in the “document root” for the instance, which is:

```
/usr/lpp/NetCommerce/html/en_US/<instance>
```

where `<instance>` is the name of your instance.

The `LIBPATH` directive tells Net.Commerce where to search for the shared objects to run. So, assume you’ve created a custom directory at HFS location:

```
/usr/lpp/NetCommerce/custom_DLL
```

The update to `LIBPATH` would be:

```
LIBPATH=/usr/lpp/NetCommerce/custom_DLL;/usr/lpp/NetCommerce...
```

You must stop and start Net.Commerce to pick up any changes to the ncommerce.envvars file.

2.5 Development checklist for the OS/390 system

Your objective should be to create an environment on the OS/390 system that is as easy and trouble free for the Windows NT or AIX programers as possible. Each time they encounter a problem on OS/390 will mean a little bit of respect lost for the platform. Therefore, we present a list to help you make sure your environment is properly set up and ready to accept your Windows NT or AIX programmers.

Make certain that:

- Your OS/390 system has TCP/IP configured and enabled.
- The Telnet service daemon is running.
- The FTP service daemon is running.
- The OS/390 server is accessible from the network on which the Windows NT or AIX developers will operate.
- Each developer has a TSO ID with an OMVS segment defined and a home directory into which they may create their development environment.
- Each developer may Telnet and FTP to the OS/390 server.
- At least one developer has the authority to stop and start the Net.Commerce server and the Web server.
- Each developer has access to SPUFI and the authority to query and update the Net.Commerce database as necessary.
- The OMVS shell `profile` used has all the environment variables set to permit the C++ compiler to be run.
- The C++ compiler is enabled, and the makefile has been tested.

Chapter 3. Develop and deploy

This chapter covers things that you need to be aware of when developing a Net.Commerce solution on NT or AIX that will be ported to OS/390. In addition to the steps outlined in Section 3.2, “Development activities” on page 16, the other sections of this chapter provide details on various aspects of planning, building, and porting your Net.Commerce applications.

The areas associated with porting your Net.Commerce application to OS/390 covered in this chapter are:

- File transfer
- Directory structure
- HTML files
- Image files
- Net.Data macros
- C/C++ programs
- DB2 database

3.1 Project skills

A Net.Commerce implementation is a complex project spanning many disciplines. The skills needed are of both a technical and business nature.

3.1.1 OS/390 skills

Since the solution will ultimately run on OS/390, skills on that platform are necessary for the project to succeed. To the extent you can find OS/390 people skilled in Net.Commerce, the better. However, even if the people doing the Net.Commerce customization are NT/AIX people, you still need to staff the project with the following OS/390 skills:

- OS/390 system administration (including SMP/E and RACF). This is to establish and maintain the OS/390 system.
- OS/390 application development (to assist with the compiling of C++ code on the 390 system, as well as helping with any integration issues with existing backend systems).
- DB2 (particularly, if you're planning on customizing the database by adding additional tables or enhancing the performance by adding indexes).

- UNIX Systems Services (because so much of Net.Commerce resides and operates within this environment).
- HTTP server (Net.Commerce uses the services of the IBM HTTP server to provide the handling of the HTTP requests).

3.1.2 AIX or Windows NT skills

You will most certainly require the skills necessary to administer the platform (NT or AIX) on which you will do your development. Some aspects of this that you will want to make sure you have are:

- System backup and restore
- Version control

The last bullet (version control) is particularly important. There is no IBM product for the OS/390 environment that does version control that would be applicable to this environment. Throughout the project, you will make many modifications to HTML, Net.Data, and C++ files. Control of that process will rely upon a system (and software) used in the NT or AIX environment.

3.1.3 Net.Commerce application skills

You will need some deep skills here, unless the Net.Commerce solution is fairly simple. You will require a team of people who have skills in:

- How the Net.Commerce command structure works.
- How to use the NCADMIN utility.
- Net.Data (which includes DB2 skills, or at a minimum, SQL skills to know how to construct the queries).
- The Net.Commerce database schema, and what information is held where.

It is unlikely you will find many people who possess these skills *and* OS/390 skills. Therefore, you will likely need a blended team.

3.1.4 Programming skills

If your plan calls for customization of the commands or overrideable functions of Net.Commerce, then you will need programming skills. The skills required are:

- C++ (for the custom commands or overrideable functions)
- Other languages (for any customization of backend systems required)

3.1.5 Web design skills

This is where the "look and feel" of the site is developed. The people involved are usually very specialized at their craft, which includes:

- Page layout and other graphic design skills
- Image composition and creation (GIF/JPG files)
- HTML
- Java and Java script, graphics, and design skills

3.1.6 Business skills

This is perhaps the most overlooked component of a Net.Commerce solution. However, the absence of these skills will most likely result in the ultimate failure of the site for all but the most simple solutions.

The decisions these people will make can significantly affect the technical design of your solution; so, it is best to involve them early in the process. You will find the different disciplines represented here will have very different views of how things should be done. Therefore, the discussions will be lively. This is why a good project manager is so important.

The business skills include:

- Project management (as stated earlier, this is a critical component of a Net.Commerce project)
- Business process knowledge (meaning someone who understands the business for which the Net.Commerce site will apply)
- Taxation (a very complicated issue, particularly since the question of how taxation on Internet sales will be handled is still under considerable debate)
- Logistics (if the site will be handling physical products, then the effect on the existing logistics system must be taken into account)
- Accounting and payment processing
- Legal (for resolution of any issues related to copyrights, trademarks, implied warranties, and to help with the taxation issue)
- Marketing (the marketing people focus on developing an image for the company; the Web site should be aligned with that image)
- Advertising (with as many "dot-com" companies out there, an unadvertised site will simply get lost in the noise)

These issues are very complex, and technical people will generally not understand all the nuances of them. Further, many customers may struggle with these issues since the Internet is a new form of doing business. Therefore, you should consider involving an IBM Global Services business consulting practice expert early in the project to assist in the formation of this strategy.

3.2 Development activities

There are many different approaches to developing a Net.Commerce application. They all share a common set of activities. These activities can, in large measure, be done on the NT/AIX platform first, with the results ported to the OS/390 server. What we'll discuss in this section are the issues to take into account while working on each of these development activities.

You do not need to completely develop the application on NT/AIX before deploying it on OS/390. Indeed, you should plan on making the development and deployment process an interactive one. (Doing so avoids the more difficult process of *migrating* the application).

Recommendation

If you have the ability to choose between developing on AIX or Windows NT, you should select the AIX platform for you development work. The AIX development environment more closely matches the OS/390 UNIX environment in such areas as syntax in commands and directory structure. As such, you will find it is easier to port your development work to the OS/390 production environment.

A typical sequence of development activities (but not necessarily one that must be followed exactly) is:

1. Install Net.Commerce environments on both platforms (they should be the same version of Net.Commerce).
2. Develop HTML pages (including images) on the NT/AIX platform and verify the layout and operation of the pages.
3. Develop and verify the Net.Data macros on the NT/AIX platform.
4. Modify the database on the NT/AIX platform to meet your solution design's needs. Document the changes made to the database so that they can be performed again on the OS/390 system (this avoids having to *migrate* the database, which is not a trivial task).

5. If you are planning to provide custom Net.Commerce commands or overrideable functions, then develop the C++ code on the NT/AIX platform and compile the DLLs.
6. To the extent possible, test the solution components on the NT/AIX platform. A complete solution test may not be possible if the solution relies on a connection to backend systems that do not exist on the NT/AIX platform.
7. Transfer the files to the OS/390 system.
8. Make modifications to the files transferred to the OS/390 server. Some aspects of HTML files, Net.Data macros, and C++ programs may need changing after to sending them to the OS/390 system:
    - Back slashes -- The NT system uses a back slash rather than a forward slash for directory specifications. If you make any direct references to directories for the NT environment using backslashes, you'll need to change them to forward slashes for the UNIX environment on OS/390.
    - Square Brackets -- The C++ code will have square brackets that will be modified during the ASCII-to-EBCDIC translation. Depending on the transfer program used, the resulting square brackets may not be correct for the C++ compiler. See Section 3.4.2, “Square brackets” on page 24 for additional information on transferring and viewing square brackets.
    - Net.Data SQL environment -- Net.Data on the NT and AIX platform utilizes the DTW_ODBC function rather than OS/390 Net.Data's DTW_SQL. Therefore, for Net.Data macros transferred to OS/390 need to be modified. See Section 3.6, “Net.Data macros” on page 27 for additional information on transferring Net.Data macros.
9. Compile C++ program on OS/390 platform.
10. Using the list of modifications to the DB2 database documented from the work performed on the NT/AIX box, make the same changes to the DB2 database on the OS/390 system.Test the ported application against the OS/390 test environment
11. Test the solution on the OS/390 system.

Testing Cycle

If you find a bug in your components (HTML, Net.Data, or C++ code), you will want to make the modifications to the files on the NT/AIX platform and then send them back to the OS/390 system. Your version control system will likely exist on that platform.

This is an interactive process, as you would imagine. The key message is this: By hosting the development activities on the NT/AIX platform, you can quite easily make the modifications. After the component has been changed and verified (whether it be an HTML file, Net.Data macro, or C++ program), it can be sent back to the OS/390 server.

The remainder of this chapter will cover things that you need to be aware of and plan for in your application development process.

3.3 Compatibility of Net.Commerce environments

As long as you utilize the Net.Commerce Version 3.1.2 on the NT/AIX platform as well as the OS/390 platform, you will insure the *functionality* of the two environments will be compatible.

However, that's not to say that the environments themselves will be necessarily the same. For instance, the custom directories in which you place your HTML, Net.Data macros, and C++ executables, might not be the same on both platforms (you should strive to make it the same, but for whatever reason, it may not be).

There are three things you should be aware of when developing your solution on the NT/AIX for deployment on the OS/390 platform:

1. The directory structures where your custom components will reside.
2. The configuration file values that determine the search paths for things, such as HTML files, Net.Data macros, and C++ executables.
3. The database entries that tell Net.Commerce about what macros to use and what executables to run when commands and tasks are called. For information on this topic, please see Section 3.8, “DB2 database” on page 40.

3.3.1 Directory structures

Because HTML files and Net.Data macros have relative references to other directories on the server, you should take care to make sure the directory structures are as close to one another as possible. Otherwise, you may find an anchor tag that worked in a HTML file developed on the NT/AIX platform suddenly doesn't work when sent up to the OS/390 server.

Some of this is influenced by the values you have coded in your configuration files (see 3.3.2, “Configuration file search path values” on page 19) because references are made *relative* to the search paths specified there. But, if you have custom directories established on the NT/AIX environment radically different from the OS/390 environment, you will need to modify the HTML and Net.Data macros to reflect the changes.

To avoid that effort, try to establish your custom directories in the same place on both platforms. Figure 2 illustrates this.

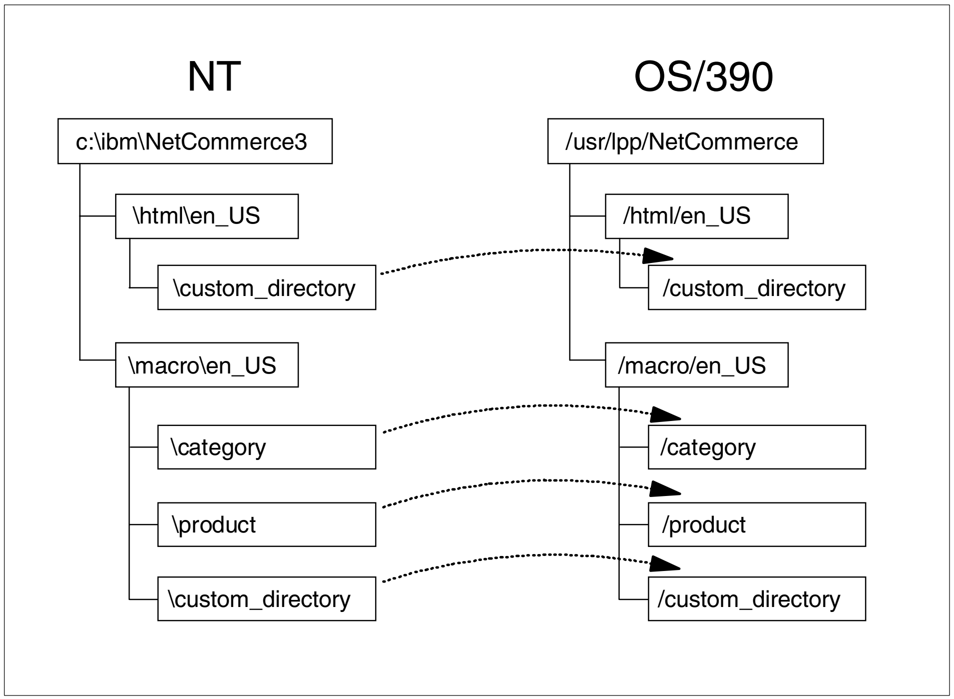


Figure 2. Common location of custom directories

This is not to say that you are bound to locating your custom directories in the locations shown in this picture. But, to the extent possible, you should strive for consistency between the environment on your NT or AIX platform and that on the OS/390.

3.3.2 Configuration file search path values

There are three configuration file parameters that determine the directories under which the Webserver or Net.Commerce will search for HTML files, Net.Data macros, or C++ executables:

- The `Pass` directives in the httpd.conf file
- The `MACRO_PATH` and `INCLUDE_PATH` directives in the db2www.ini file
- The `LIBPATH` directive in the ncommerce.envvars file

The directories under which the searches will be conducted are done on a *relative* basis.

3.3.2.1 Pass directives in httpd.conf file

The `Pass` directives in the Webserver's configuration file (typically `httpd.conf`) tell the Web server how to map the directory specification on the URL to the actual directory on the server. This function exists because, without it, URLs would become prohibitively long and cumbersome.

A typical URL, calling a static HTML page, is comprised of the components illustrated in Figure 3. We are very carefully specifying "static HTML page" for this illustration because a URL containing a Net.Commerce *command* consists of many other components.

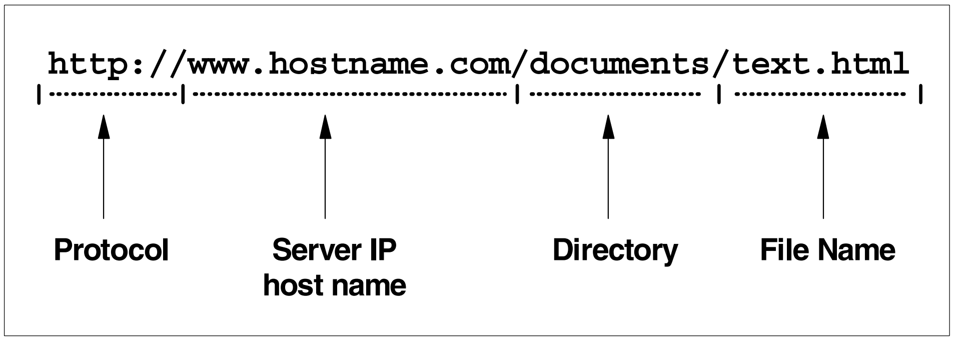


Figure 3. Components of static HTML URL

The directory specified on the URL may be read `documents`, but the actual directory in which the file `text.html` resides might be something much longer, for example:

```
/usr/lpp/NetCommerce/html/en_US/custom/documents/text.html
```

To require a browser user to type out all of that would be asking too much of the user. So, rather than having a user type all of that out, the Web server has a `Pass` directive, which maps the much shorter documents to the longer, actual directory name:

```
Pass /documents/* /usr/lpp/NetCommerce/html/en_US/custom/documents/*
```

What this tells the Webserver is this: Any time you receive a URL with a directory request of `documents` with anything after it, then map that request to the *actual* directory of /usr/lpp/NetCommerce/html/en_US/documents. Then, search for the file name requested in that directory.

The Webserver may have any number of `Pass` directives specified; so, you can map short names to quite a few actual directory locations.

Here's where the relative mapping comes into play. Let's say the URL that's received is:

```
http://www.hostname.com/documents/today/text.html
```

Recall that the `Pass` directive said "any time you receive a directory request of `documents` with *anything after it*, then map the request to the actual directory of /usr/lpp/NetCommerce/html/en_US/documents. In this example, what comes after the directory `documents` is `/today/text.html`. Notice that `/today` is itself a directory. The directory `/today` is a relative offset from the specified `Pass` directive that consists of `/documents`.

Your HTML files (as well as Net.Data macros) will consist of many of these relative offsets. Anchor tags (links) included in your HTML will not typically specify the entire URL including the host name. To include the host name would make the HTML file usable by that Webserver only, which is not a good thing. Rather, the HTML will consist of `/documents/today/text.html` designations. It is up to the `Pass` directives in the httpd.conf file to resolve those to the correct *actual* directory locations.

Therefore, whereas it is not necessary to have the exact directory structure between the development platform on NT/AIX and the OS/390 system, the `Pass` directives must be able to resolve the *relative directory references* to the correct directories. Otherwise, you'll be spending a lot of time changing HTML files.

Having a directory structure that is very similar between the two platforms will mean you won't have to think as hard when making sure the `Pass` directives are correct.

3.3.2.2 MACRO_PATH and INCLUDE_PATH in db2www.ini file

When a Net.Data macro is called, Net.Data uses the directories found on the `MACRO_PATH` directive of db2www.ini to know where to search for the macro. Include files contain commonly used pieces of Net.Data macros that are dynamically "included" into a macro with a `%include` statement. Net.Data searches the `INCLUDE_PATH` in the exact same manner as it does for `MACRO_PATH`. We'll focus only on `MACRO_PATH` for the rest of this section.

Let's assume the `MACRO_PATH` statement looks like this:

```
MACRO_PATH /usr/lpp/NetCommerce/macro/en_US
```

If you were to call a macro with only the name of the macro, say `testpage.d2w`, Net.Data would search the /usr/lpp/NetCommerce/macro/en_US directory for that file. If it was not there, you would get a Net.Data error.

Note

You may have any number of directories specified on `MACRO_PATH`; so, don't think you have to put all your macros in one directory.

Now, let's say you had some subdirectories under the directory named on `MACRO_PATH`. Net.Data would *not* search into those subdirectories for a macro unless the subdirectories were explicitly named on `MACRO_PATH`.

However, you can make Net.Data go into those subdirectories by providing a relative directory offset when you call the macro. For example, consider the directory structure shown in Figure 4 on page 22:

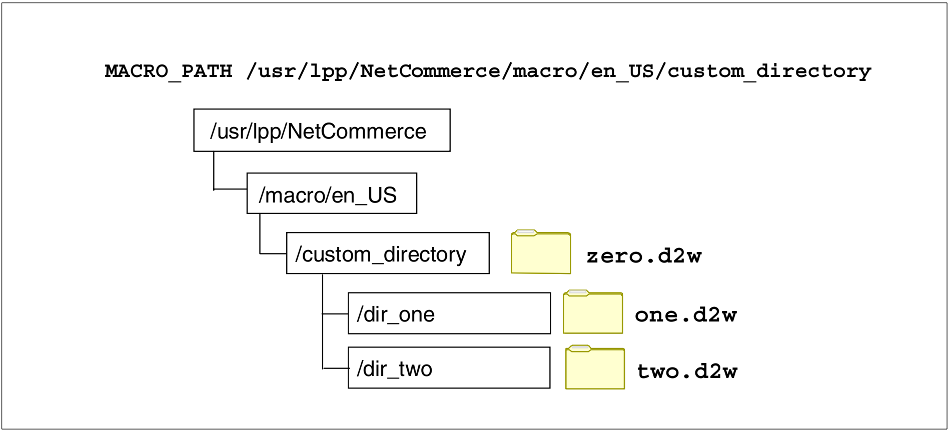


Figure 4. Net.Data macros found in subdirectories

Assume an `ExecMacro` command to execute the macro:

```
http://<host>/cgi-bin/ncommerce3/ExecMacro/zero.d2w/report
```

Because the `zero.d2w` macro is located in the directory specified on `MACRO_PATH`, the macro will be found. But, the same command used to execute `one.d2w` will *not* work:

```
http://<host>/cgi-bin/ncommerce3/ExecMacro/one.d2w/report
```

The reason is the `one.d2w` macro is one directory down in the structure. But, you can reference it with a relative path offset on the command:

```
http://<host>/cgi-bin/ncommerce3/ExecMacro/dir_one/one.d2w/report
```

If the solution you develop on the NT/AIX platform makes reference to relative paths, then you must make sure the relative paths will resolve properly on the OS/390 system. This is yet another reason to strive to keep the directory structures similar between the development and target system.

3.3.2.3 LIBPATH in ncommerce.envvars file

The `LIBPATH` directive in the ncommerce.envvars file tells Net.Commerce where to look for the *.so files (shared object files) to load for the commands and overrideable functions. See Section 2.4.3, "Custom Net.Commerce command and OF directory" on page 11.

For a custom command or overrideable function to be found and loaded by Net.Commerce, they *must* be located in the directories specified on `LIBPATH`. There is no concept of relative paths with `LIBPATH`; the *.so file is either found, or it is not.

By the way, Net.Commerce loads all the commands and overrideable functions when Net.Commerce is initialized. Therefore, if you modify a command or overrideable function, or add one, you must restart Net.Commerce to pick up the change.

3.4 File transfer issues

Once you develop a component on the NT/AIX platform, you need to get it to the OS/390 system. The most common method of doing this is to use File Transfer Protocol (FTP). To use FTP, you need to have a workstation with TCP/IP and FTP, and the OS/390 server will need the same.

Another method of transferring file to OS/390 is to use the IND$FILE transfer mechanism with your 3270 emulator. This is *much* less common than FTP.

There are two issues you need to be aware of:

1. ASCII to EBCDIC translation
2. Square brackets

3.4.1 ASCII to EBCDIC translation

Years ago, IBM defined EBCDIC (Extended Binary-Coded Decimal Interchange code) as its character encoding scheme (8-bits for a character). The American National Standards Institute (ANSI) defined a different code called ASCII (American National Code for Information Interchange) that uses seven bits for a character. All UNIX and PC systems use ASCII in one form or another, but IBM OS/390 continues to use EBCDIC.

What this means is that some files created on the NT or AIX platform (in ASCII) will need to be translated to EBCDIC to be used on the OS/390. Fortunately, FTP will do this for you when you transfer a file to OS/390 using FTP if you specify the ASCII option. Some files should *not* be translated, and you can accomplish that with the BINARY option.

Table 1shows what file transfer option you should use for the different file types.

Table 1. File transfer options for various file types

File Type	**Transfer Option**
HTML files	ASCII
GIF/JPG files	Binary
Net.Data macro files	ASCII
C++ source code	ASCII

3.4.2 Square brackets

Square brackets (keyboard characters [and]) may become an issue when developing your C code. At issue is the EBCDIC hex value of the characters on the OS/390 system. The C/C++ compiler is expecting a certain hex value for those characters. It’s possible to generate the wrong hex values if you use a 3270 emulator to either transfer a file containing square brackets or type the characters into a file in a TSO session.

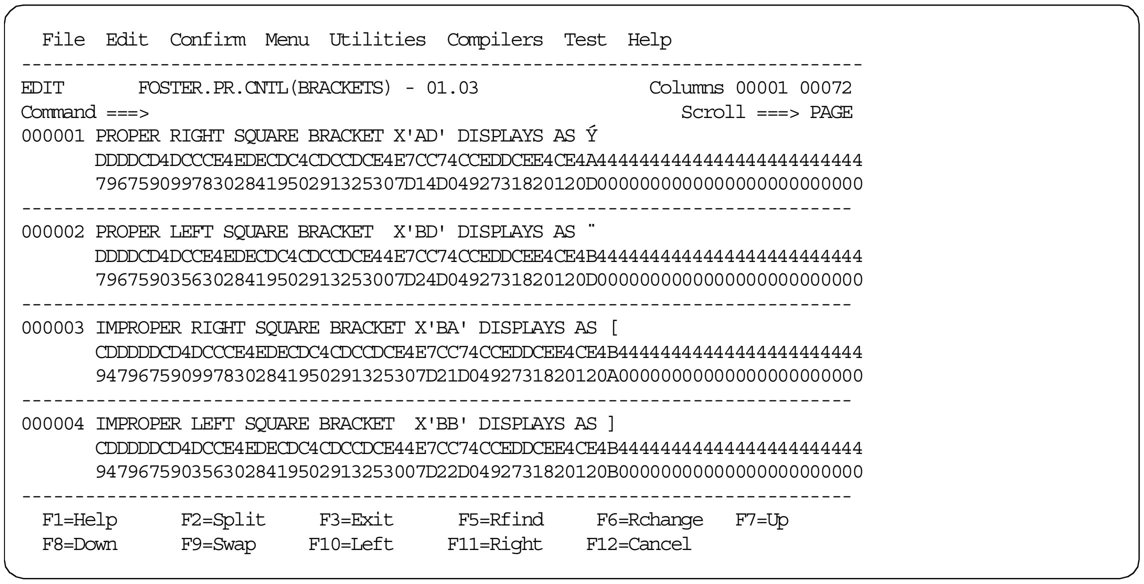
Table 2 provides the hex values for square brackets in ASCII and EBCDIC.

Table 2. Square bracket hex values

Character	**ASCII hex**	**Proper EBCDIC hex**	**Improper EBCDIC hex**
Left Square ([ )	x’5B’	x’AD’	x’BA’
Right Square ( ])	x’5D’	x’BD’	x’BB’

Square brackets in files sent to the OS/390 system with the FTP ASCII option will result in the proper x’AD’ and x’BD’ EBCDIC characters on the OS/390 system. The C++ compiler will accept them.

If you transfer the file to the OS/390 system using the 3270 emulator package’s transfer utility, you’ll probably end up with the improper x’BA’ and x’BB’. The C++ compiler will flag them as in error.



```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       FOSTER.PR.CNTL(BRACKETS) - 01.03               Columns 00001 00072
Command ===>                                                 Scroll ===> PAGE
000001 PROPER RIGHT SQUARE BRACKET X'AD' DISPLAYS AS Ý
       DDDDCD4DCCCE4EDECDC4CDCCDCE4E7CC74CCEDDCEE4CE4A4444444444444444444444444
       796759099783028419502913253O7D14D0492731820120D0000000000000000000000000
-----------------------------------------------------------------------------
000002 PROPER LEFT SQUARE BRACKET  X'BD' DISPLAYS AS ¨
       DDDDCD4DCCE4EDECDC4CDCCDCE44E7CC74CCEDDCEE4CE4B4444444444444444444444444
       796759035630284195029132530O7D24D0492731820120D0000000000000000000000000
-----------------------------------------------------------------------------
000003 IMPROPER RIGHT SQUARE BRACKET X'BA' DISPLAYS AS [
       CDDDDDCD4DCCCE4EDECDC4CDCCDCE4E7CC74CCEDDCEE4CE4B44444444444444444444444
       9479675909978302841950291325307D21D0492731820120A00000000000000000000000
-----------------------------------------------------------------------------
000004 IMPROPER LEFT SQUARE BRACKET  X'BB' DISPLAYS AS ]
       CDDDDDCD4DCCE4EDECDC4CDCCDCE44E7CC74CCEDDCEE4CE4B44444444444444444444444
       9479675903563028419502913253007D22D0492731820120B00000000000000000000000
-----------------------------------------------------------------------------
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
  F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 5. 3270 display of square bracket characters

You can avoid any problems with square brackets easily enough if you transfer the files using FTP to the OS/390 system. But here’s the twist. If you FTP the code to the OS/390 system, and then use a 3270 emulator to view the file, you won’t see square brackets. You’ll see Ý and ¨ that looks like a a capital Y with an accent mark over it and a double-quote symbol. A 3270 emulator ISPF text edit session in hex mode, as shown in Figure 5, displays the proper and improper characters and their ECBDIC values. The square brackets are fine; the 3270 emulator is just displaying them using different characters.

If you view the file contents with a Telnet or Rlogin session, the square brackets will look just fine.

What you *don’t* want to do is overtype the characters with the 3270 emulator. This will result in the *improper* x’BA’ and x’BB’ EBCDIC characters being placed in the file. The 3270 emulator will show them as "[" and "]", but the C++ compiler will flag them as errors.

The bottom line is this: Use FTP to transfer to the OS/390 system the files with square brackets. If you use a 3270 emulator to view the file, be aware the brackets will look funny, but in reality, they are just fine. And, don't use the 3270 emulator to overtype the characters. This will result in the wrong characters being placed in the file.

3.5 HTML and image files

HTML and image files are perhaps the most portable of all the components of a Net.Commerce solution. The OS/390 Webserver does not require anything different or special in either the HTML files or the image files; to it, they're just files to be passed to the browser.

Excellent HTML generation tools exist on the NT and AIX platforms; so, you should take advantage of them when designing and building your pages. You could hand-code the HTML files using the OS/390 editors, but, in reality, most people take advantage of the tools that exist on NT/AIX and transfer the HTML up to the OS/390 system.

Image files (GIF and JPG) are binary representations of a picture, and as such, are almost universally portable across platforms. Excellent tools exist for the development of image files on AIX and NT as well (not to mention the Apple MacIntosh systems); so, the development of these files will very likely be done on the lower platform as well.

One consideration to remember is that HTML files in the OS/390 UNIX System Services Hierarchical File System (HFS) are, by default, stored in EBCDIC format. When the OS/390 Web server passes these files down to the browser, it translates them from EBCDIC to ASCII "on the fly".

Note

It is possible to store HTML files in the OS/390 HFS in ASCII format and have the Web server pass them down to the browser untranslated. See *WebSphere Application Server for OS/390 HTTP Server Planning, Installing, and Using,* SC31-8690, if you wish to store your files in ASCII format in the HFS.

Image files are stored in the HFS system as binary objects. No EBCDIC to ASCII translation occurs when the file is sent to the browser.

Therefore, when you transfer your HTML files and image files from the distributed development platform to the OS/390 platform, you will you need to transfer them using the proper option to translate the files to EBCDIC during the file transfer process. Which transfer option you should use for HTML and image files is provided in Table 1 on page 24.

3.5.1 Things to watch out for with HTML files

You need to be aware of the following items while developing your HTML pages on the distributed platform.

3.5.1.1 Case sensitivity

The OS/390 UNIX Systems Services is case sensitive (as is AIX); so, care will have to taken to ensure that files are stored in their correct case in the OS/390 file system. The Windows NT system is not case sensitive; so, it's possible for a mis-match in case between a file reference in an `<a href>` tag and the actual file name; Net.Commerce on NT will not care. This is not the case on OS/390. OS/390 UNIX Systems Services is case sensitive, and these files need to be matched correctly to work properly.

3.5.1.2 Relative file references

HTML files and Net.Data macros will usually reference image files and other HTML files using a relative file reference rather than a fully qualified location. The reference will either be for the file name alone or with one directory appended to the file name. This is done to keep the file portable. It relies on the Web server or Net.Data to resolve the actual location of the file with the information provided in each program's configuration files.

If your directory structure is different between the NT/AIX platform and the OS/390 platform, you may need to update the configuration files or change the HTML files so that the files are resolved properly. See 3.3.1, "Directory structures" on page 18 for a discussion of this issue.

3.6 Net.Data macros

Net.Data capability and macro format is very similar between the different platforms. However, there are some differences, and you need to be aware of these and take these differences in to account as you port your Net.Data macros to OS/390.

3.6.1 Net.data functional comparison

The degree of common function between the platforms on which Net.Data runs is impressive. However, you should be aware that there are environment variables and functions that are specific to the Windows NT or UNIX platform that do not apply to OS/390 or are not part of Net.Data for OS/390. Table 3 lists the Windows NT or Unix platform functions that you should avoid using in your Net.Commerce applications.

Table 3. Net.Data differences

Category	What's not supported for OS/390
Environment variables	DATABASE DTW_EDIT_CODES DTW_PAD_PGM_PARMS LOGIN NULL_RPT_FIELD PASSWORD
Miscellaneous variables	DTW_DEFAULT_MESSAGE DTW_LOG_LEVEL
Web registry functions	There are no Net.Data Web registry functions that apply to OS/390.

If you utilize any of the items listed in Table 3 in your Net.Data macros, they will not port cleanly to the OS/390 environment. You will then need to modify the macros to work on your OS/390 production server. If you want to learn more about the functions listed in Table 3, you can find details about them in the Net.Data Reference manual found at the following IBM Web site:

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

3.6.2 Things to be aware of regarding Net.Data macros

When developing your Net.Data macros on the NT or AIX platform, keep the following issues in mind.

3.6.2.1 File transfer option

Net.Data macros are stored in the OS/390 HFS as EBCDIC files. When you create them on the NT or AIX platform, they will be in ASCII format. Therefore, you must make certain that the ASCII-to-EBCDIC translate option is in effect when you FTP the files to the OS/390 system. See 3.4, “File transfer issues” on page 23 for a complete discussion of this.

3.6.2.2 Forward and backward slashes

If you are migrating from Windows NT, you need to change the DOS-type paths (for example, `c:\IBM\NetCommerce\macro`) to UNIX type paths (for example, `/usr/lpp/NetCommerce/macro`). Typically, this means that backslashes need to be changed to forward slashes in the macros and the drive letters be removed.

Note

The back/forward slash issue is one of the reasons that it is recommended that you perform your develop activity on AIX instead of the Windows NT platform. When you develop your HTML pages and Net.Data macros on the AIX platform, they directly, easily and quickly port to the OS/390 Unix System Services environment.

If you are developing your Net.Commerce application on Windows NT, you can use the UNIX `sed` command to change characters in files. For example, to change backslashes in a file called macro_nt.d2w, and write it to a file called macro_390.d2w, the command is:

```
sed ‘s:\\:\ /:’ <macro_nt.d2w> macro_390.d2w
```

Note

The < and > are literal; so, you need enter them into the command to effect the correct change.

3.6.2.3 Change DTW_ODBC to DTW_SQL

The Windows NT and UNIX versions of Net.Commerce generally use Open Database Connectivity (ODBC), but the OS/390 version of Net.Commerce does not. DTW_ODBC is supported on OS/390, but it doesn’t perform as well as DTW_SQL; so, it’s not the default. Therefore, after you transfer your Net.Data macros to the OS/390 server, you must change all occurrences of DTW_ODBC to DTW_SQL in each Net.Data macro.

You can use the UNIX `sed` command to globally change DTW_ODBC to DTW_SQL in a file. The `sed` command to make a global change in a file looks like this:

```
sed ‘s/DTW_ODBC/DTW_SQL/’ <macro_nt.d2w> macro_390.d2w
```

To change all files in a directory, you need to make a UNIX shell loop, for example:

```
for a in *
do
cp $a $a.0
sed ‘s/DTW_ODBC/DTW_SQL/’ < $a.0 > $a
done
```

3.6.2.4 Avoid using tabs

Some people like to use tabs for formatting macros on Windows NT. Unfortunately, this can cause problems when the macro is transferred to the OS/390 server. The PC tab character does not translate correctly when it goes thought the ASCII-to-EBCDIC process. In addition, tabs are not supported by the ISPF editor on OS/390.

If your macro has tabs in it, they'll appear as "hidden characters" in the OS/390 copy of the macro. These characters are identified by the ISPF editor using a `find p'.'` command. The hexadecimal representation for the tab is `x'05'`. They should be changed to spaces or nulls before the macros are executed.

3.6.2.5 Avoid using new lines

OS/390 Net.data macros should not contain the `\n` or new line character. If the `\n` character is found, Net.Data on OS/390 splits the statement within a language token, and, as a result, Net.Data may not accept it. For example:

```
<INPUT TYPE=hidden NAME=url VALUE="/cgi-bin/ncommerce3/Order\n
     Display?status=P&merchant_rn=$(MerchantRefNum)">
```

should be changed to:

```
<INPUT TYPE=hidden NAME=url VALUE="/cgi-bin/ncommerce3/
     OrderDisplay?status=P&merchant_rn=$(MerchantRefNum)">
```

3.6.3 Testing the Net.Data macros

After the Net.Data macros have been sent to the OS/390 server, you should test them to make certain they function as you wish. The easiest way to test the macros is to invoke them directly using the `ExecMacro` command. For example, the command to test a macro called `ported.d2w` is:

```
http://<host_name>/cgi-bin/ncommerce3/ExecMacro/ported.d2w/report
```

where `<host_name>` is the IP host name of your OS/390 server. See Section 3.9.2, "Net.Data macros" on page 43 for more information on testing Net.Data macros.

3.7 C/C++ programs and application development

Up to this point, we have focused on the HTML and Net.Data aspects of Net.Commerce customization. This can carry you a good way towards your objective of customizing your site. However, there may be some special function you're looking for that's not supplied with the product. Depending on what you wish your Net.Commerce site to do, you may need to customize the command or tasks used by Net.Commerce. This involves C++ programming.

Note

Coding custom C++ programs is not strictly required for Net.Commerce. Many customers limit their customization to HTML and Net.Data macro changes. The advantage to custom C++ is complete flexibility of function as well as performance. C++ programs will generally perform better than Net.Data macros.

3.7.1 Commands, tasks, and overrideable functions

In Net.Commerce, a command is an instruction to Net.Commerce to do some work. The command is actually comprised of a smaller unit of work called a task. The number of tasks associated with a command varies by command

Figure 6 illustrates how Net.Commerce commands relate to tasks and how tasks relate to the overrideable functions (OFs).

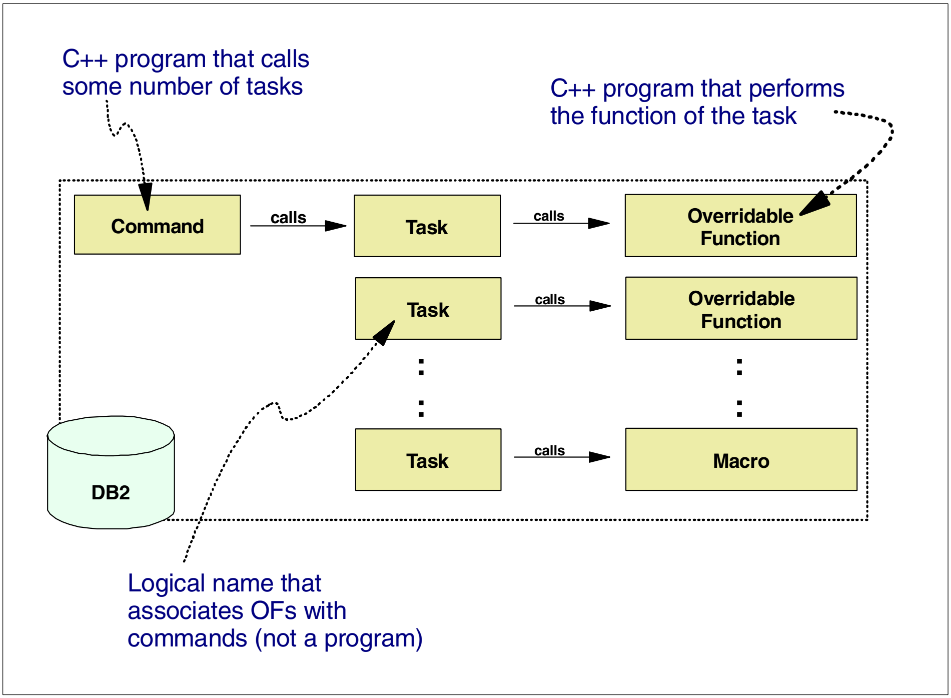

C++ program that calls some number of tasks
C++ program that performs the function of the task
Command
calls
Task
calls
Overridable Function
Task
calls
Overridable Function
Task
calls
Macro
DB2
Logical name that associates OFs with commands (not a program)

Figure 6. Commands, task, and overrideable functions

Commands and overrideable functions are programs that are executed. Tasks are nothing more than symbolic names related to overrideable functions. Commands call the tasks and Net.Commerce determines which overrideable function is to be run based on the task called. In Net.Commerce for OS/390, commands and overrideable functions are implemented in C++ shared objects. The equivalent on NT or AIX is a Dynamic Link Library (DLL).

3.7.2 Where to develop, compile, and run

The development model we're proposing in this document involves first developing and compiling the C++ code on the NT/AIX workstation. When the programs have been verified for clean compilation and unit operation, the source code is sent to the OS/390 system where it will be *compiled again*. Final testing is performed on the OS/390 system, which would include functional testing as well as integration and performance testing as well.

This process is illustrated in Figure 7.

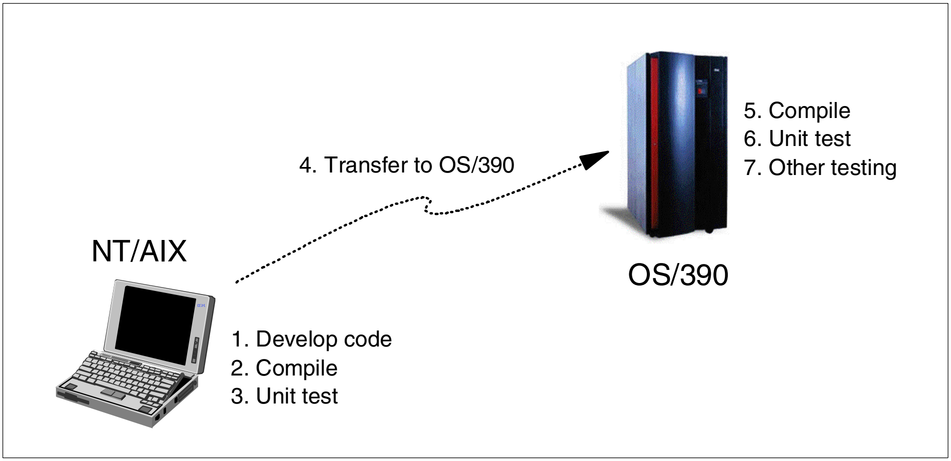


Figure 7. Overview of development model

The amount of testing you can accomplish on the NT or AIX platform may be limited by the degree to which that platform can access the backend systems to which your solution will integrate. See Chapter 4, “A real life example” on page 47 for an illustration of a real-life project where testing on the NT/AIX platform was restricted.

3.7.3 NT Visual C++ development

With Net.Commerce V3.1.2, Net.Commerce commands and overrideable functions are written in C++. In the Windows NT environment, Microsoft Visual C++ is often the tool used to create and compile the source files.

Compilation of Net.Commerce C++ code on Windows NT can also be achieved using the `make` command that comes with the Visual C++ compiler. The input to the make command is a `makefile`, which provides instructions to the make command about how to build the DLL and what the code dependencies are.

Remember

The compiled DLLs on the NT/AIX workstation can not simply be transferred to the OS/390 system. The C++ source code must be set up and re-compiled.

When compiling a Net.Commerce DLL on Windows NT, you will see many error messages from the link process. This is expected and is due to the dynamic nature of the runtime environment for dll's, where resources the dll needs are not available until it is loaded and running in the Net.Commerce environment.

3.7.4 AIX and OS/390 command line development

The OS/390 UNIX Systems Services shell interface is very similar to the UNIX shell interface any UNIX programmer is already familiar with. This is intentional because it allows UNIX programmers a very easy transition to OS/390 UNIX Systems Services.

The three most common ways in which to start an OS/390 UNIX System Services shell are:

1. rlogin
2. telnet
3. TSO OMVS command

Which one you use is really a question of what kind of environment you're looking to approximate most closely. A programmer unfamiliar with OS/390 will probably not like the 3270 interface of the TSO OMVS shell. For those people, the rlogin or telnet approach is probably better.

If you plan on using rlogin or telnet, it will be necessary to enable those services on the OS/390 TCP/IP stack. The UNIX programmers will wish to use the vi editor to modify files. To use vi on OS/390, you first have to export a valid TERM type in your rlogin session, as the default term type of ANSI is not supported by vi on OS/390. An example of the command you need to enter before using vi is:

```
export TERM=vt100
```

Once the rlogin or telnet session has successfully started, traditional UNIX commands can be used in the shell.

For OS/390 programmers who don't like the vi editor (and there are many who fit that description), the TSO OMVS shell or ISHELL can be used to access HFS files. The OMVS shell is a command line interface that tries to approximate (as best as 3270 can) a traditional UNIX command shell. It requires a pretty good grasp of UNIX commands to use effectively. The ISHELL, on the other hand, is an ISPF panel interface that is used to copy, move, browse, modify, and edit HFS files.

To access the OMVS or ISHELL interface on TSO, you need a 3270 emulator on your workstation.

3.7.5 OS/390 compilation

On OS/390, you use the `cxx` command to compile C++ source code. More typically, you will actually use the `make` command, which provides a non-graphical batch front end to the `cxx` command. As discussed previously, the `make` command is also available on Windows NT; so, application developers should be familiar with it.

The `make` command takes, as input, a makefile, which provides instructions to the `make` command about how to build the shared object file and what the code dependencies are.

3.7.6 OS/390 makefile issues

Creating a valid makefile for OS/390 to compile your ported C++ source can be a time consuming process. To assist you in building a working makefile for OS/390, we offer the sample makefile in 3.7.7, “OS/390 sample makefile” on page 35.

Key points in this makefile are:

- Compiler option LANGLVL(EXTENDED) *must* be used

- DLL is specified as a link option.
- Includes for the Language Environment (LE) header files and Net.Commerce include files to successfully build a Net.Commerce shared object (known as a DLL on Windows NT) on OS/390.

You need to customize the makefile to your OS/390 environment by making the following changes:

- Change **NC_ROOT** to your Net.Commerce installation root.
- Change **DB2PATH** to where you installed DB2 JDBC (if using JDBC).
- Change **LE_HLQ** to the high-level qualifier for your Language Environment Data sets.
- Change **TARGET_DLL** to the name of the DLL (shared object) you are building.
- Change **TARGET_DATASET_NAME** if you want to store the DLL in an OS/390 data set.
- Change **PDS** to the member name of the DLL (within TARGET_DATASET_NAME) CC = cxx.

3.7.7 OS/390 sample makefile

The following is a sample makefile to compile Net.Commerce C++ source on OS/390:

```
NC_ROOT = /usr/lpp/NetCommerce
DB2PATH = /usr/lpp/db2/db2510
LIBPATH = $(NC_ROOT)/lib/
LE_HLQ = PP.ADLE370.OS390R7D
MVSINC = -I"//'$(LE_HLQ).SCEEH.H'" \
-I"//'$(LE_HLQ).SCEEH.ARPA.H'" \
-I"//'$(LE_HLQ).SCEEH.NET.H'" \
-I"//'$(LE_HLQ).SCEEH.NETINET.H'" \
-I"//'$(LE_HLQ).SCEEH.SYS.H'" \
-I"//'SYS1.SFOMHDRS'"
INCLUDE = -I$(NC_ROOT)/adt/include \
-I$(NC_ROOT)/adt/include/common \
-I$(NC_ROOT)/adt/include/containers \
-I$(NC_ROOT)/adt/include/database \
-I$(NC_ROOT)/adt/include/messages \
-I$(NC_ROOT)/adt/include/objects \
-I$(DB2PATH)/include \
-I/usr/include \
-I/usr/include/sys \
-I../common \
-I/usr/include/arpa \
-I/usr/include/netinet \
-I//$(LE_HLQ).SCEEH.H \
-I//$(LE_HLQ).SCEEH.ARPA.H }
-I//$(LE_HLQ).SCEEH.NET.H \
-I//$(LE_HLQ).SCEEH.NETINET.H \
-I//$(LE_HLQ).SCEEH.SYS.H \
-I//SYS1.SFOMHDRS
# ---------------------------------------------------
# Component specific information
# ---------------------------------------------------
TARGET_DLL = miles
CNAME = $(TARGET_DLL).cpp
ONAME = $(TARGET_DLL).o
SONAME = $(TARGET_DLL).so
SOXNAME = $(TARGET_DLL).x
SIDEDECKS = $(LIBPATH)/nc2util.x \
$(LIBPATH)/server_objs.x \
$(LIBPATH)/nc3_common.x \
$(LIBPATH)/nc3_containers.x \
$(LIBPATH)/nc3_dbc_db2.x \
$(LIBPATH)/nc3_messages.x
# ---------------------------------------------------
# MVS specific stuff
# ---------------------------------------------------
TARGET_DATASET_NAME = MY.SCMNLMOD
PDS = $(TARGET_DATASET_NAME)(TMPLTCMD)
# ---------------------------------------------------
# Rules
# ---------------------------------------------------
all : $(SONAME)
$(SONAME): $(ONAME)
@echo Linking
@rm -f $(LIBPATH)/$(SONAME) $(LIBPATH)/$(SOXNAME) $(SONAME)
$(SOXNAME)
cxx -Wl,DLL -L $(LIBPATH) -o ./$(SONAME) $(ONAME) $(SIDEDECKS)
@echo Copying target to $(PDS)
cp ./$(SONAME) '//$(PDS)'
touch $(SONAME)
$(ONAME) :
@rm -f $(ONAME)
@echo Compiling
cxx -+ -O -DAIX -D_ALL_SOURCE -DMVS -D_OE_SOCKETS \
-W0,LANGLVL(EXTENDED),EXPORTALL \
$(INCLUDE) \
-c $(CNAME)
clean :
@echo Cleaning
rm -f $(ONAME)
rm -f $(LIBPATH)/$(SONAME) $(LIBPATH)/$(SOXNAME) \
$(SONAME) $(SOXNAME)
```

3.7.8 OS/390 C++ coding issues

This section covers the key coding issues that you need to keep in mind as you develop your Net.Commerce application so that they will easily and smoothly port from your development environment to the OS/390 server.

3.7.8.1 Porting NT/AIX C++ code to OS/390 UNIX Systems Services

If you develop the C++ code to ANSI standards, then it should be a straight-forward process of sending the code to OS/390 and recompiling. If you do things in your code, such as direct manipulation of ASCII characters, then you will have to scrub that for the EBCDIC environment on OS/390 unless you use the LIBASCII library. Section 3.7.8.3, “ASCII-like application environment” on page 37 gives more information on using the LIBASCII library.

For information on porting code from UNIX/NT to 390, have a look at the UNIX Systems Services porting Web page at:

`http://www.s390.ibm.com/oe/bpxa1por.html`

The porting guide itself is a PDF file that can be downloaded from:

`http://www.s390.ibm.com/ftp/os390/oe/docs/port.pdf`

and contains a full discussion of the ASCII/EBCDIC counteractions, plus many more. This topic will be explored further in this redbook.

3.7.8.2 Differences in compiler output

On the Windows NT or AIX platform, a compiled executable will take the form of a DLL, which is a single file with an extension of `*.dll`. That DLL will represent either a custom command or an overrideable function. On OS/390, the custom command or overrideable function will have two components: A *.so file (the shared object code) and a *.x file (the side deck).

3.7.8.3 ASCII-like application environment

If your C++ code is written on the NT or AIX platform to be ASCII dependent, there is a way to use the code without re-writing it for OS/390. A library called LIBASCII (`http://www.s390.ibm.com/oe/bpxa1toy.html`) has been developed for use by programmers who are accustomed to ASCII programming. Thus, if a program is compiled to ASCII using this layer, all parameters and return values for character based system calls, for example, *setenv() and printf(),* will be translated to and from EBCDIC as required.

Note

Compiling to ASCII is a capability provided by the OS/390 C/C++ compiler.

When this feature is used, all string literals are encoded as ASCII values in the executable image, thus preserving all ASCII properties, such as ‘a’ >‘A’, etc., and negating the need to rewrite ASCII dependent code. Literals will, however, require translation to EBCDIC or ASCII when they are, for example, returned from or passed to an OS system call or third party product. Character type return values from system calls will also require translation to ASCII.

The ASCII layer provides this facility. Using this approach, if the code executes correctly on an ASCII-based operating systems, it should execute correctly on an EBCDIC-based operating system with all ASCII dependencies intact. This could be of great help if complicated and undocumented sections of ASCII dependent code are encountered.

For some restrictions with the usage of the LIBASCII tool, please refer to Section 6 of the IBM Whitepaper, *Porting a UNIX Application to OS/390 UNIX - Problems Encountered and Lessons Learned*. This document is currently available at: `http://www.s390.ibm.com/oe/aixdb2/aixdb2.html`

3.7.8.4 Where to put your executables

Your custom commands and overrideable functions should reside in a location separate from the default Net.Commerce executable library. Doing this will avoid confusion about which executables are custom and which are from Net.Commerce, and it avoids the possibility of having your custom executables wiped out by a service release from IBM. We illustrated how to establish isolated directories for executables in 2.4.3, “Custom Net.Commerce command and OF directory” on page 11.

You may optionally copy your C++ commands and overrideable functions from the HFS to OS/390 data sets. There is, perhaps, a slight performance advantage to this, but it’s never been measured. The sample makefile in 3.7.7, “OS/390 sample makefile” on page 35 shows you how to do this. Also, before OS/390 2.5 and the OS/390 USS `extattr` command, having code in OS/390 data sets was the only way to ensure that they were APF-authorized. All that being said, most Net.Commerce for OS/390 customers run their commands and overrideable functions directly from the HFS and not from OS/390 data sets.

If you do copy the executable to an MVS dataset, you will still need an entry in the HFS directory. The *.x file will reside in the HFS directory, and the *.so file will need to be an *external link* to the member in the MVS data set containing the code.

An external link is a special UNIX file type that points to a member in an MVS data set. When that UNIX file is referenced, OS/390 automatically relates the request over to the MVS dataset. The UNIX process requesting the file knows nothing of this going on behind the scenes. To create an external link, you would execute the following OS/390 UNIX shell command:

```
rm <mycode>.so
```

where `<mycode>.so` is the UNIX file name of the shared object for your custom command or overrideable function. This removes the file. The next step would be to create the link:

```
ln -e MYCODE <mycode>.so
```

where `MYCODE` is the name of the member in the MVS data set in which your compiled code is stored. This command will create the external link, which will look like a file, but have a "type" of `syml` (as viewed in ISHELL).

The OS/390 data set containing the MYCODE member must be referenced by one of the following:

- On the LIBPATH variable of the <instance_name>.envvars file and the ncommerce.envvars file
- On the Net.Commerce startup procedures `STEPLIB DD` statement
- In the OS/390 system link list or LPA

Placing the data set name on the `LIBPATH` statement is the recommended approach.

3.7.8.5 Typical problem areas

Some of the typical problem areas that one should watch out for are:

- Hard-coded ASCII characters in C code as well as shell scripts
- Using the high-order bit of a character for some special purposes
- Assuming the alphabet (‘a’......’z’) is continuous, this is true in ASCII but not in EBCDIC encoding. For example, there are special characters that appear in the EBCDIC sequence between ‘a’ and ‘z’.
- Using code generated by *lexx* or *yacc* (the code needs to be regenerated on OS/390).

- Applications that talk to remote systems via sockets (such as an ftp client).
- Code that relies on byte order of data may not be portable. (PC systems are “little endian”, that is, the left most byte is the most significant, while OS/390 and most UNIX systems are “big endian”.
- In ASCII, when comparing the character “A” to the character “a”, “A” < “a”, but in EBCDIC, “a” < “A”.

Some of these issue can be worked around by using the LIBASCII library. This was discussed in 3.7.8.3, “ASCII-like application environment” on page 37.

The document, `http://www.s390.ibm.com/products/oe/bpxa1p03.html`, can be referenced for further information.

You may want to reference Section 4.3, “Lessons learned and overall project assessment” on page 59 that lists other items learned in a customer implementation.

3.8 DB2 database

Net.Commerce uses the DB2 database to access, manipulate, and store its data and logical structures. While the name DB2 may be common on the different platforms, Database Management System and operating system differences mean Net.Commerce implementations can vary from platform to platform. By building identical database environments on both the development and deployment platforms, you can avoid any problems associated with these differences.

Note

You can see the database schema and information in the tables for a Net.Commerce system using the online help information provided with Net.Commerce. This online help can be accessed via the browser by specifying the following URL:

```
http://<host_name>/nchelp/
```

3.8.1 Propagating changes to the database

The database format for Net.Commerce is *essentially* (though not exactly) the same between Windows NT, AIX, and OS/390. The approach you need to follow when working with the data base on the development system and the OS/390 deployment system is to:

- Make certain you start from the same point on both platforms. This means you should start with a new database on both platforms, and make sure you load the same skeleton sample database on each.
- As changes are made on the NT/AIX platform, make the same changes on S/390. This applies both to content changes as well as schema changes. And, always document the changes you make to the database.

What you’re looking to avoid is constructing the customized database completely on the NT/AIX platform and then face migrating the database to OS/390. Migrating the database involves using the DB2 UNLOAD and LOAD utilities. It is better to simply make sure the changes made on the NT/AIX platform are documented and then made on the OS/390 system fairly soon thereafter.

3.8.2 Registering custom commands and OFs

When you create a custom command or overrideable function and compile the executable, you need to register that executable in the Net.Commerce database. The CMDS and OFS tables hold information on the commands and overrideable functions for Net.Commerce.

The online help for Net.Commerce provides information on how to register the command or overrideable function. For example, the online help provides this sample SQL for registering a custom overrideable function:

```
insert into ofs (refnum , dll_name, vendor, product,
   name, version, description)
values ((select max(refnum) from ofs) + 1, '< DLL_Name>',
 '<IBM>' , '<NC>' , '< Overridable_Function_Name>',<1.0> , < description> )
```

Unfortunately, there is a "nested select" statement that does not work on OS/390 DB2 V5.1. Essentially, the `(select max(refnum) from ofs)+1` portion of this is looking for the maximum value of `refnum` from the OFS table, and then setting the `refnum` value for the insert to that value + 1.

To work around this, you will need to make this a two step, manual process:

- Manually select the maximum REFNUM value from the OS/390 table using the SQL:

  ```
  select max(refnum) from <owner>.OFS;
  ```

  This will return the highest value found for REFNUM in the OFS table.
- Add one to the maximum value found and then manually change the SQL and update the OS/390 table:

```
insert into ofs (refnum , dll_name, vendor, product,
   name, version, description)
values (XXXXX, '< DLL_Name>', '<IBM>' , '<NC>' ,
  '< Overridable_Function_Name>',<1.0> , < description> )
```

Where `XXXXX` is the `max(refnum)+1` value you computed by hand.

The same process needs to be used when registering custom commands as well.

Note

Do not assume the REFNUM values found in the NT/AIX database are the same as those found in the OS/390 database. It is important that you query and register the commands and overrideable functions on the OS/390 platform without assuming anything based on what you saw in the NT/AIX database.

3.9 Testing considerations

Throughout this document, we make the point that this development process is really a *co-development* process on NT/AIX and OS/390. The benefit in developing the components on the distributed platform is the flexibility of the inherent in the platform. The benefit of running the solution on OS/390 is the reliability, scalability, and security inherent in that platform.

Whereas it is possible to develop the entire solution on the NT/AIX platform and porting (or *migrating*) it at some later date to the OS/390 platform, that is not what we're advocating here. To develop the entire solution on the NT/AIX platform will require, if backend systems are accessed, architecting and developing into the solution distributed integration designs. Such distributed integration designs may not be the best when everything is located on a single server, such as OS/390.

Therefore, the components are developed on the distributed platform and tested to the fullest extent possible on the NT/AIX box. But, the extent of testing will probably not be as complete as is required once the complete solution is constructed on the OS/390 platform.

Here, we'll discuss some testing considerations for the various components of the solution.

3.9.1 HTML files

HTML generation tools are plentiful and of very good quality on the NT and AIX platforms. Therefore, they make a very good environment on which to develop and test the HTML files. These can be relatively self-contained and can be tested fully on the distributed platform. The things you want to check for are:

- The look and layout of the page
- The inclusion of all the intended graphics
- The operation of all the anchor tags and form actions on the pages

As discussed in 3.3.2.1, “Pass directives in httpd.conf file” on page 20, the references under these anchor tags and form actions will likely be in the form of relative offsets. Therefore, while they may function properly on the NT/AIX platform, they may not on the OS/390 if the directory structure doesn’t mirror that on the NT/AIX box. Some testing on the OS/390 box will, therefore, be required.

3.9.2 Net.Data macros

Net.Data macros are, by their design, intended to get information from a source outside the macro. For Net.Commerce, this typically implies an SQL call to the database. Therefore, other components need to be available on the system to effectively test a macro.

The development and testing of a Net.Data macro on the NT/AIX platform can be fairly comprehensive. It would include insuring the following is correctly coded in the macro:

- The format of the page, its visual appeal, and the inclusion of all referenced graphics.
- The proper query and return of the desired information from the DB2 database.
- The proper handling of anticipated error conditions.
- The proper calling of the specified macro by a given task of Net.Commerce (which, incidentally, verifies the proper registration of the named macro in the MACROS table of Net.Commerce).

When these are ported up to the OS/390 platform, you should retest them to make sure that:

- The connection to the database is made (although you don’t need to check this for every macro; if one macro works, then all will since the connection to DB2 is made at a more global level than the individual macro).
- The SQL in the macro queries for, and returns, the proper information (this verifies not only the SQL on the OS/390 platform, but also verifies that the information in the database is what was expected).
- The links and buttons on the resulting page work properly (this tests that the relative offsets referenced in anchor tags and form actions map properly to the directory structure on the OS/390 system; see 3.3.2.2, “MACRO_PATH and INCLUDE_PATH in db2www.ini file” on page 21 for a discussion of how the relative offsets are mapped).
- The proper macros are called when a given Net.Commerce view task is invoked (this verifies the proper registration of the macro in the MACROS table).

3.9.3 Database changes

Throughout the construction of your Net.Commerce solution on the NT/AIX platform, you will make changes to the DB2 database (see 3.8.1, “Propagating changes to the database” on page 40).

3.9.3.1 Types of changes to database

The type of changes you will likely make to the database are:

- The initial creation and loading of the skeleton database (Net.Commerce comes with several different sample databases; you should create the same one on both platforms)
- Mall and merchant information
- Product and category content (most people develop a mass import script that loads all this information up into the NT/AIX database; that script is then sent to the OS/390 system and used with mass import to achieve the same result)
- Registration of macros to view tasks

And, the changes you may make depending on the level of complexity of your solution are:

- Registration of custom commands and overrideable functions into the database
- Additional table indexes
- Additional tables

3.9.3.2 Testing of the database changes

There are two basic approaches to testing the database changes:

1. Visual inspection after the changes have been made (this is why documenting the changes made on the NT/AIX platform is so important; it provides a *checklist* of things to visually inspect)
2. Observation of the behavior of the Net.Commerce solution (Net.Commerce's behavior is, in large measure, controlled by the content of certain tables of the database; if you expect macro XYZ to run to show the shopping cart, for example, and some other macro run, then you know you have a problem)

There is no magic here: It is manual work.

3.9.4 C++ programs

The degree to which you can test the C++ programs you develop is very much dependent upon what those programs do. For example, if they make a CICS call to get some data, and CICS doesn't exist on the NT/AIX platform, then that aspect of the C++ program can't be tested. However, that's not to say that other aspects of the C++ program can't be tested.

The most basic testing that you accomplish by compiling on the NT/AIX platform is that the syntax of the C++ code is correct. In addition to this, any use of Net.Commerce SQL class structure to get information from the Net.Commerce database can be tested on the NT/AIX platform. Further, the correct invoking and passing of parameters to and from the C++ code can be checked.

When the code is sent to the OS/390 system it must be recompiled and placed in the proper Net.Commerce LIBPATH directories (see 3.3.2.3, "LIBPATH in ncommerce.envvars file" on page 23). Net.Commerce must be restarted to pick up the changes because the commands and overrideable functions are loaded up at initialization.

Full testing of the command and overrideable function can be accomplished on the OS/390 system. Remember that the commands and overrideable functions are invoked based on the registration of the command and overrideable function in the database. Therefore, it is important that any changes to the database be made on the OS/390 platform as well (see 3.9.3, "Database changes" on page 44).

If the OS/390 system on which you do your initial testing is not the intended production system (or a mirror of it), then you may not have the subsystems and backend systems to which the C++ code is intended to connect. In those cases, you may be forced to develop "stubs" to emulate the backend systems so that as complete a test of the solution can be accomplished. See Section 4.2.4.1, “Accessing the OS/390 system” on page 56 for a case study on a real-life customer situation where this was necessary.

Chapter 4. A real life example

This chapter tells of a real-life story of a Net.Commerce implementation done first on NT, then deployed up on an OS/390 system. The implementation involved a real customer, but the details presented in this chapter are about the implementation and not the customer. For our discussion, we will call that customer “Company XYZ”.

Those involved with this real-life project worked through many of the issues discussed in this redbook. Therefore, in describing how they went about the project, we will be able to illustrate many of the points presented in the earlier chapters.

4.1 Background and history

Company XYZ invited IBM to discuss its Net.Commerce product and evaluate how well it provided what XYZ needed for its new business venture. Historically, XYZ distributed its product through wholesalers and now wanted to offer its products directly to the customer over the Internet. XYZ was very skilled in OS/390 and wanted to utilize the capacity on its existing S/390 systems for this project.

This mapped nicely to Net.Commerce OS/390.

But, there was a twist: Company XYZ had a very large database on DB2 that contained much of the information about its customers, pricing, taxation and inventory levels. To the extent possible, they wanted to avoid duplicating that information in the Net.Commerce database.

This was possible with Net.commere OS/390 because of the customizable framework of the product. Custom commands and overrideable functions (OFs) could be written to access the information. All that was required was a team from IBM who knew how to code in C++ and develop the custom commands and overrideable functions.

But wait, there was one last twist in the story: Company XYZ did not wish to allow the IBM team to directly access its database. Rather, XYZ wished to use custom COBOL programs (a programming language for which they had deep skills) to access the data in their DB2 database. This was a perfectly understandable desire, and meant the Net.Commerce custom commands and overrideable functions would call the COBOL programs to do the data access.

This was still within what could be done with Net.Commerce. So, the architecture of the solution looked like what is shown in Figure 8.

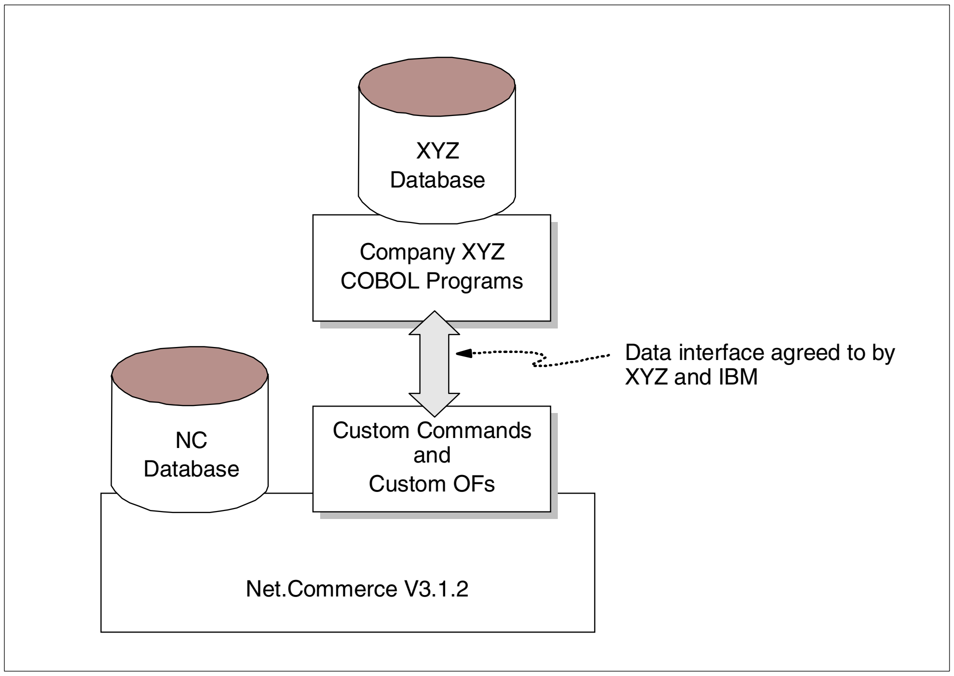


Figure 8. Company XYZ's Net.Commerce architecture (very high level)

There was, as one would expect, quite a bit more detail to the project than has been shown in Figure 8. These architecture issues, in some cases, turned out to be quite complex. But, to discuss them here would be beyond the scope of this book. Many are related to the real “Company XYZ” environment and not to the development and deployment concepts being presented here.

4.2 The project begins

IBM Global Services was engaged to perform the customization work required for this project. The first task was to locate skilled people for the team, then they could get to work designing and developing the solution.

4.2.1 In search of skills

A search was done to locate people within IBM Global Services (IGS) who were skilled in both Net.Commerce customization as well as OS/390. That is a fairly unique combination, and the search yielded virtually no results.

There were plenty of Net.Commerce experts in IBM Global Services, but their platform of expertise was NT or AIX. Conversely, there were plenty of OS/390 experts in IBM Global Services, but their skills were focused on other things, such as OS/390 systems administration, DB2, application development, and OS/390 performance tuning.

The project time line didn't permit the NT/AIX Net.Commerce specialists to learn OS/390 (which would have been a long learning curve in any event). And, since Company XYZ's requirements suggested a highly customized Net.Commerce solution, the time line didn't permit the OS/390 specialists to learn Net.Commerce at that level of detail either.

The solution was to use the Atlanta e-business Design Center for the customization of the Net.Commerce environment. The strategy behind the "e-center" approach is fairly simple:

- Concentrate your skills in a single geographical "center".
- Bring the customer's work to the center and draw upon the concentration of skills present at the center.

4.2.2 Creating the development environment

Once the e-business Design Center had been selected, the question of what development environment to use came to the surface. The project team faced two opposing facts:

- Company XYZ's environment was OS/390.
- The Net.Commerce experts in the e-center were familiar with NT and AIX, and did not know OS/390 at all.

So, the options open to the team were these:

- Do all the development on an OS/390 system. This would include all the HTML, Net.Data, and the C++ development through 3270 or Telnet sessions to a 390 host.
- Do much of the development on either NT or AIX, then transfer the source up to the OS/390 test system and validate the components there before shipping the solution off to Company XYZ.

The first option was certainly feasible, but the developers in Atlanta wished to take advantage of the flexibility in coding on a workstation. In addition, the e-center had a version control mechanism in place that operated at the workstation level. Therefore, the second option was chosen: Develop on NT, validate and deploy on OS/390.

4.2.2.1 The development model

The development process for the project is shown in Figure 9.

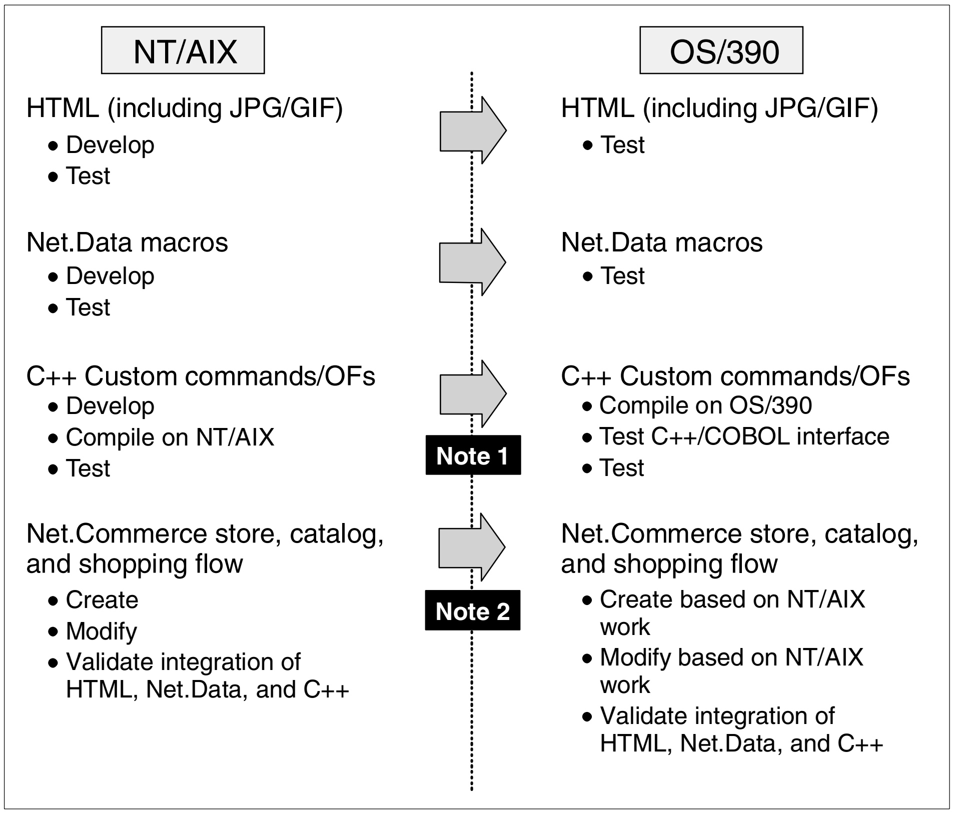

NT/AIX
OS/390
HTML (including JPG/GIF)
• Develop
• Test
HTML (including JPG/GIF)
• Test
Net.Data macros
• Develop
• Test
Net.Data macros
• Test
C++ Custom commands/OFs
• Develop
• Compile on NT/AIX
• Test
Note 1
C++ Custom commands/OFs
• Compile on OS/390
• Test C++/COBOL interface
• Test
Net.Commerce store, catalog, and shopping flow
• Create
• Modify
• Validate integration of HTML, Net.Data, and C++
Note 2
Net.Commerce store, catalog, and shopping flow
• Create based on NT/AIX work
• Modify based on NT/AIX work
• Validate integration of HTML, Net.Data, and C++

Figure 9. Customer XYZ development model

There are two key points to be noted about the development model shown in Figure 9:

Note 1: The Windows NT or AIX environment was going to be used for something more than simply an editor. The code was going to be compiled (to validate syntax) and tested to the extent possible. Remember, that a key piece of the architecture was an interface to COBOL programs written by Company XYZ, which did not run on the Windows NT or AIX environment. Note also that this model required the C++ source code be recompiled on OS/390. You can't just ship the compiled executable up to OS/390 and run it.

Note 2: This is *not* to say the Net.Commerce solution was developed on Windows NT or AIX and *migrated* to OS/390. Migrating the Net.Commerce database from NT or AIX to OS/390 is a non-trivial task (see *Migrating Net.Commerce Applications to OS/390*, SG24-5438). The development model called for the store to be created and the database to be populated on Windows NT or AIX and validated. Then, the steps taken to do these things would be *done again* on OS/390. The Mass Import utility was used to automate the catalog aspect of this. The other steps were documented carefully and re-executed on OS/390.

4.2.2.2 Sourcing the OS/390 system

The Atlanta e-business Design Center did not have an S/390 machine on which to do the validation, nor did they have S/390 skills even if they had one. To overcome this, an OS/390 environment was established in Gaithersburg, Maryland. The developers in Atlanta would access the Gaithersburg system as depicted in Figure 10. The developers used were able to use FTP to transfer source code, Telnet to get to a UNIX prompt on OS/390, and a 3270 emulator to get to an ISPF screen for basic OS/390 work, such as running SPUFI to do SQL work and starting/stopping the Net.Commerce server.

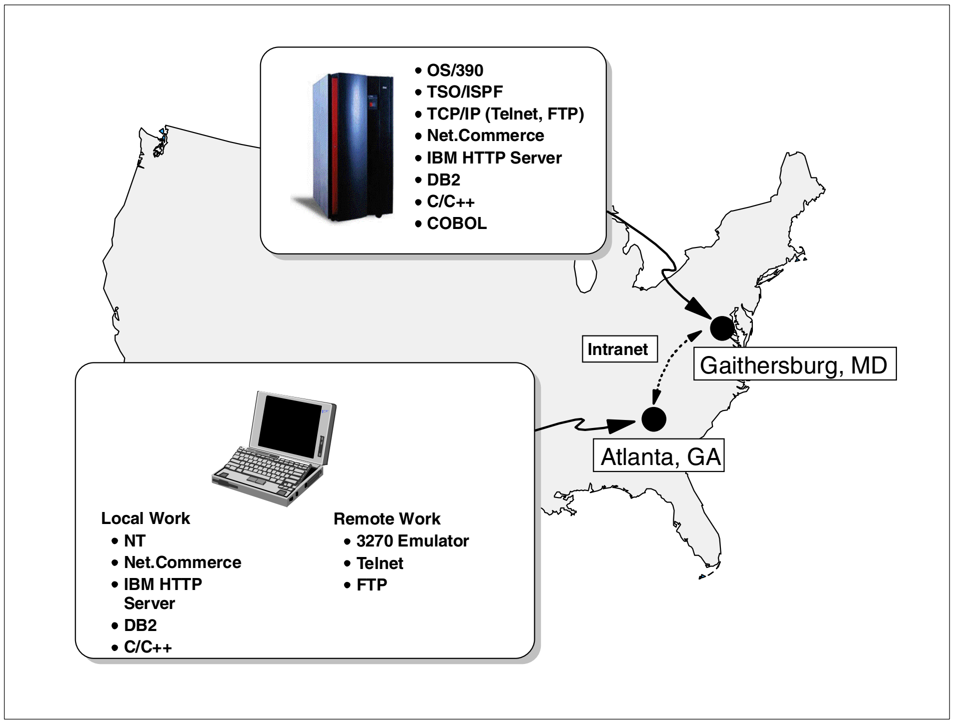


Figure 10. Project XYZ development environment

4.2.2.3 Isolating Company XYZ development on the OS/390 system

The OS/390 system on which the Atlanta developers would do their work was also used for other Net.Commerce activities, such as classroom education. It was important to provide an isolated environment for the Atlanta developers. This was so their work wouldn't be stepped upon by others and that they wouldn't inadvertently affect other efforts.

Here is a summary of the steps taken to isolate the Company XYZ's efforts on this system:

- Each developer in the Atlanta e-business Design Center was provided a TSO ID with its own HFS home directory. This TSO ID was used when logging in with a 3270 emulator as well as when they FTP-ed code up to the OS/390 system or accessed the machine with Telnet.

  The home directories were /u/xyz01, /u/xyz02 ... /u/xyz05. It was in subdirectories under these home directories that the developers placed their C++ source code and executed the compiler.

- A separate Net.Commerce “instance” was configured for the Atlanta developers. This permitted them to have a separate copy of Net.Commerce that they would be free to start and stop as needed.

  This instance’s configuration files were kept in the unique directory created for this instance:

  ```
  /usr/lpp/NetCommerce/html/en_US/atlanta
  ```

- A separate Net.Commerce database was created and loaded for the Atlanta developers. This permitted them to make modifications to things, such as the commands, overrideable functions and macros registered to the system. The sample “Demomall” database was created and loaded. It was called atlanta.
- A separate HFS directory for HTML files, and a separate HFS directory for JPG/GIF image files. These directories were referenced with `Pass` directives in the Web server’s httpd.conf file:

  ```
  Pass /xyz/* /usr/lpp/NetCommerce/html/en_US/atlanta/xyz_html/*
  Pass /xyz_images/* /usr/lpp/NetCommerce/html/en_US/atlanta/xyz_images/*
  ```

- A separate HFS directory for Net.Data macros. The directory was referenced on the `MACRO_PATH` and `INCLUDE_PATH` directives of the db2www.ini file for the instance of Net.Commerce used by the IBM Global Services team in Atlanta:

  ```
  MACRO_PATH /usr/lpp/NetCommerce/macro/en_US/xyz_macro;/usr/lpp...
  INCLUDE_PATH /usr/lpp/NetCommerce/macro/en_US/xyz_macro;/usr/lpp...
  ```

- A separate HFS directory for the compiled custom command and custom overrideable function “shared object” (*.so) files. The directory was referenced on the `LIBPATH` directive of the ncommerce.envvars file:

  ```
  LIBPATH=/usr/lpp/NetCommerce/html/en_US/atlanta/xyz_bin:/usr/lpp...
  ```

The environment that was established, which looked like the structure shown in Figure 11 on page 54.

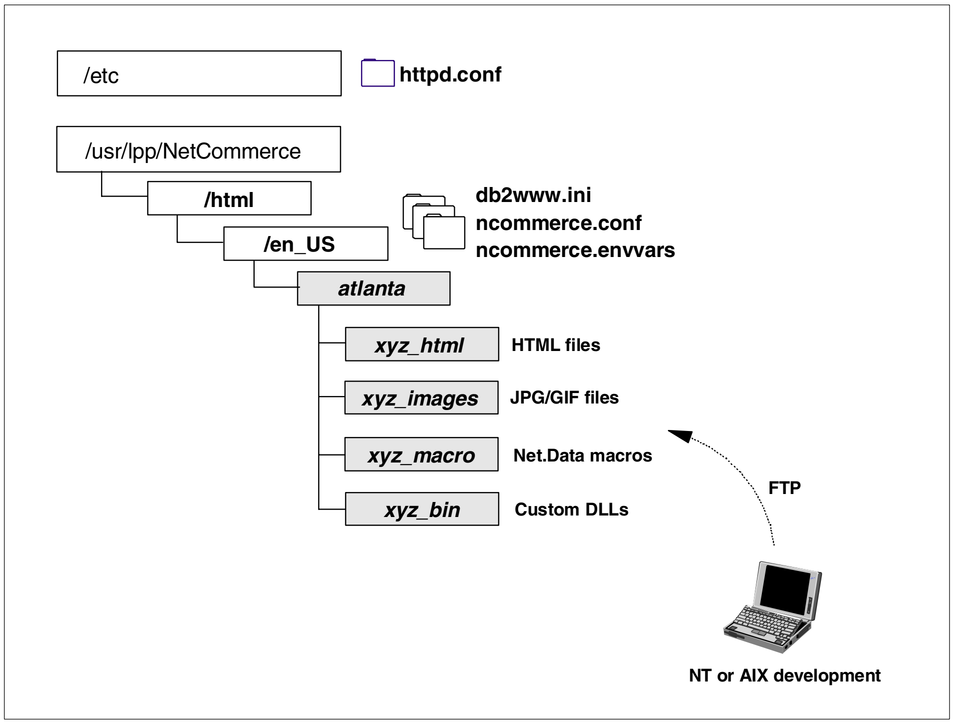


Figure 11. Environment after isolated HFS directories created

4.2.2.4 Providing a COBOL program interface

The final solution would involve COBOL programs written by Company XYZ that would be called by the Net.Commerce custom C++ code. The COBOL programs would be the ones who accessed Company XYZ's backend database. This design was decided upon so that the IBM developers didn't have to code directly to this database. Company XYZ's database was very large and could not easily be cloned to the Gaithersburg OS/390 system.

So, we had a dilemma. How could the Atlanta developers test their C++ code's behavior against the data interface if they didn't have the COBOL programs and database to work against? The solution to this was to have Company XYZ develop COBOL "stubs" that emulated the defined interface but didn't actually access a database (it simply echoed back the same dummy data each time, regardless of the parameters that were received). The development and test solution architecture is shown in Figure 12 on page 55.

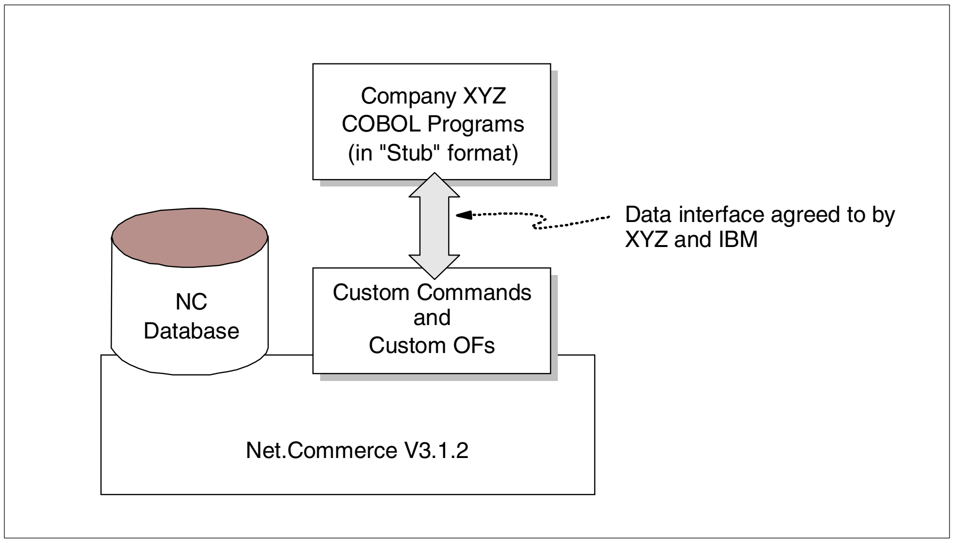


Figure 12. Development and test solution architecture

Note

The architecture solution shown in Figure 12 is not the final solution but only represents the environment for developing and testing the solution.

4.2.3 Start up jitters

The first time one of the developers attempted to compile some C++ code on the S/390 system, they encountered some problems. These problems were related to the environment variables for the OMVS session shell not being set correctly, and the makefile not being correct.

This redbook is not intended to be a reference for how to set up the C/C++ environment on OS/390. Oversights like this occur in any project, and this project was no different.

This does serve to illustrate a key point: Though it is possible to have NT/AIX Net.Commerce developers largely shielded from the OS/390 specifics, it is impossible to do an OS/390 engagement without OS/390 skills on the team. In this example, an OS/390 application developer familiar with compiling code on OS/390 should have been involved earlier to establish that aspect of the environment.

4.2.4 Issues encountered

This section contains the things we encountered during the development project. We list them here so that you can be aware of them before you start your project and avoid them.

4.2.4.1 Accessing the OS/390 system

When the project first started, the instructions provided the developers in Atlanta described how to log on to TSO and navigate to the OMVS shell or ISHELL. That was all fine and good, but the 3270 interface is an acquired taste and one not easily acquired for someone familiar with NT or AIX. Telnet access had been established from the start, but it had not been part of the instructions; so, the developers didn't know it was available.

You will find that UNIX programmers unfamiliar with OS/390 will be pleasantly surprised when they Telnet into an OS/390 box. The developers in Atlanta said it best: "Hey, this looks just like a UNIX box!" When a programmer Telnets into OS/390, they get a UNIX prompt, a TTY terminal type, and the VI editor, which are things they are familiar with and accustom to using.

Advertise this aspect of UNIX Systems Services. It will ease any apprehension about the UNIX interface.

4.2.4.2 ASCII-to-EBCDIC conversion

This is the one aspect of UNIX Systems Services that isn't "just like a UNIX box." Despite documenting the need to do ASCII-to-EBCDIC conversion, we still ran into minor problems with this. Watch out for JGP/GIF binaries in particular. Many FTP packages default to ASCII mode, which will result in the conversion of the binary JPG/GIF file into EBCDIC. If this happens, the JPG/GIF file won't be usable. The symptom you'll see on the browser is the "missing image" symbol. Yet, when you look in the HFS directory, the JPG/GIF file exists. The problem is that when the EBCDIC conversion occurred, it's no longer a usable JPG/GIF file.

Table 4 contains a quick reference of what requires conversion and what does not.

Table 4. Quick reference on ASCII-to-EBCDIC conversion

Type of File	ASCII-to-EBCDIC?
HTML files	Yes (see note)
JPG/GIF image files	No
Net.Data macros	Yes

Type of File	ASCII-to-EBCDIC?
C++ source code	Yes
Note: Purists will argue that it's possible to upload HTML files in binary format so that they are stored on the OS/390 server as files in ASCII format. They would be correct: The HTTP Server on OS/390 has a setting that allows HTML files in the HFS to be treated as ASCII format and served over the wire to the browser without the usual EBCDIC-to-ASCII conversion. But, that's not the default, and unless you know for certain that's how the Web server is set up, the HTML files will require conversion to EBCDIC when you FTP them to the OS/390 box.	

Watch out for square brackets in your C++ code. Transferring square brackets from an ASCII workstation to an EBCDIC host has been an issue for years because of differences in the EBCDIC code pages for the square bracket values. See Section 3.4.2, "Square brackets" on page 24 for more details on this.

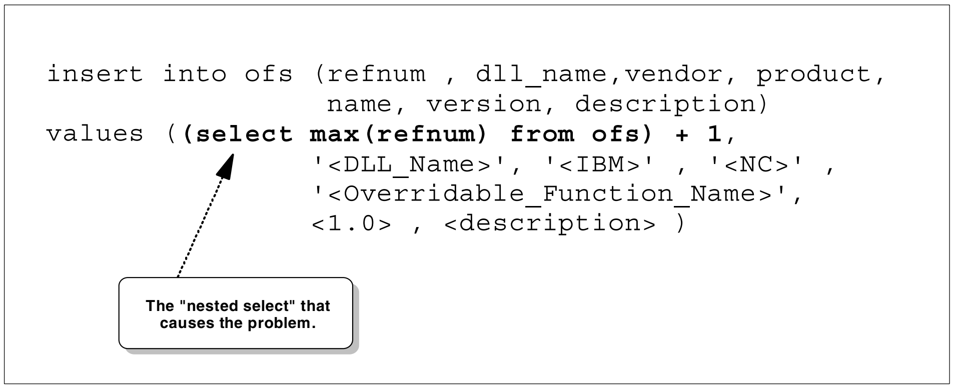
4.2.4.3 Using SPUFI

The Atlanta developers were familiar with the DB2 interface of NT or AIX (a handy graphical interface). SPUFI was a bit of a departure from the Windows NT or AIX model. First of all, it required the developers to log on to the TSO session, which is something they had never done before. Further, it required them to navigate the ISPF editor that SPUFI uses for the creation of the SQL.

A possible solution to this would have been to use DB2 Connect on the NT machine. That would have permitted SQL queries to be issued on the NT box, but executed against DB2 on the OS/390 system. But our time line was short; so, we simply provided instruction on how to use SPUFI. The developers picked it up quickly.

4.2.4.4 SQL differences

We encountered a minor problem when the developers transferred to the OS/390 box the SQL they used to register their commands and overrideable functions into the database. It turns out the example SQL provided in the online help for Net.Commerce (even the online help on OS/390) uses "nested select statements" to dynamically determine the next reference number to use. The sample nested SQL statement provided in the online help is shown in Figure 13 on page 58. Unfortunately, nested select statements aren't supported in DB2 V5 on OS/390.



```
insert into ofs (refnum , dll_name,vendor, product,
                  name, version, description)
values ((select max(refnum) from ofs) + 1,
                 '<DLL_Name>', '<IBM>' , '<NC>' ,
                 '<Overridable_Function_Name>',
                 <1.0> , <description> )
```

Figure 13. Example SQL found in Net.Commerce online help

The way we got around this (and the way you will have to work around it as well) is to make this a two-step process:

1. Query the OFS table to determine the maximum value for the REFNUM column. This is done with this SQL:
   `select max(refnum) from <owner>.OFS;`
2. Take the number you receive from this query, add one to it, and then manually update "insert" SQL so that the value for refnum inserted into the table is that number.

This isn't pretty, but it works.

The other minor issue we encountered is that when using SPUFI you must supply the owning RACF ID name on the tables referenced in the queries. This is an OS/390 requirement, so someone familiar with Windows NT or AIX DB2 needs to be told what the owning ID is and how to structure the SQL syntax to include the ID name. An example of this would be:

```
select max(refnum) from cmnsrv.OFS;
```

In this case, `cmnsrv` is the RACF ID that owns the database. The default ID is `cmnsrv`, which you are instructed to create when you install Net.Commerce.

4.2.4.5 Coordinating the COBOL/C++ interface

Figure 8 on page 48 showing the solution architecture contains a data interface between the C++ programs being written by the IBM Global Services developers in Atlanta and the COBOL programs being written by the developers in Company XYZ.

Early in this project, we advised developers at both Company XYZ and the Atlanta e-business Center that if there was one area to pay very close attention to, and to document as tightly as possible, this data interface was it. The reason is that if developers on either side of the interface modified their code in a way that changed the data format, the behavior could change. These are difficult problems to debug.

But, people will be people, and we did encounter a few problems in this area. One in particular was when the developers at Company XYZ removed a SYNC parameter from the COBOL code. They forgot to advise the Atlanta developers of this. The removal of the SYNC parameter meant that a `s9(9) COMP` value in the COBOL program did not align the data on a full word boundary, and the data structure collapsed.

This book isn't intended to be a COBOL or C++ programming primer. We're offering this issue from the XYZ project as an illustration of why the original advice is so important. (We're *not* illustrating this to pass judgment or lay blame). When you customize Net.Commerce to interface with other processes, it's absolutely critical to pay close attention to that interface.

4.3 Lessons learned and overall project assessment

The major lessons learned were these:

- It's not possible to do an OS/390 Net.Commerce project without any OS/390 skills. What's being advocated in this redbook is that the core team responsible for the deep Net.Commerce customization doesn't need to be OS/390 people. But, OS/390 people are clearly needed elsewhere, as was described in Section 1.2, "Team structure" on page 2.
- If custom commands or overrideable functions are to be written, review the recommendations made in Section 3.6.2, "Things to be aware of regarding Net.Data macros" on page 28 to avoid committing the design to something that won't work or work well on OS/390.
- It's really beneficial to have an OS/390 application programmer involved if custom Net.Commerce C++ programs are being written. The OS/390 application programmer doesn't need to be completely fluent in Net.Commerce custom C++. But, if they team with Windows NT or AIX programmers who are, it makes the project run more smoothly.
- It's possible to mask a good deal of the OS/390 environment from Windows NT or AIX developers by utilizing Telnet and FTP. If the Windows NT or AIX developers are interested in learning some of the OS/390 operational tasks and procedures (as was the case with the XYZ project

team from Atlanta), then things, such as TSO, SPUFI, and starting/stopping products can be opened up to them.

Overall, the project turned out to be successful because of the teaming that occurred between the OS/390 people and the Windows NT or AIX people. This included a willingness to learn some aspect of each other’s area of expertise.

Perhaps the most important thing was that early on in the project everyone set aside the competitive “my platform is better than your platform” arguments that sometimes plague cross-teaming projects. Instead, the focus was on delivering a successful solution to Company XYZ. That was the recipe for success.

Appendix A. Special notices

This publication is intended to help project leaders and technical specialists to understand how to develop Net.Commerce applications on the NT or AIX platform for porting and deployment on the OS/390 platform. The information in this publication is not intended as the specification of any programming interfaces that are provided by Net.Commerce for OS/390. See the PUBLICATIONS section of the IBM Programming Announcement for Net.Commerce for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate

Appendix B. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

B.1 IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 69.

- *Migrating Net.Commerce Applications to OS/390*, SG24-5438
- *Building e-commerce Solutions with Net.Commerce: A Project Guidebook,* SG24-5417
- *Integrating Net.Commerce with Legacy Applications*, SG24-4933
- *Net.Commerce for OS/390*, SG24-5154
- *Secure Electronic Transactions: Credit Card Payment on the Web in Theory and Practice*, SG24-4978
- *Exploring Net.Commerce Hosting Server*, SG24-5505

B.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at http://www.redbooks.ibm.com/ for information about all the CD-ROMs offered, updates and formats.

CD-ROM Title	**Collection Kit Number**
System/390 Redbooks Collection	SK2T-2177
Networking and Systems Management Redbooks Collection	SK2T-6022
Transaction Processing and Data Management Redbooks Collection	SK2T-8038
Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
AS/400 Redbooks Collection	SK2T-2849
Netfinity Hardware and Software Redbooks Collection	SK2T-8046
RS/6000 Redbooks Collection (BkMgr)	SK2T-8040
RS/6000 Redbooks Collection (PDF Format)	SK2T-8043
Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

B.3 Other resources

These publications are also relevant as further information sources:

- *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862
- *Porting Applications to the OpenEdition OS/390 Platformtitle*, GG24-4473
- *IBM Commerce Integrator User's Guide for AIX*, SC09-2878
- *IBM Commerce Integrator User's Guide for NT*, SC09-2865
- *Net.Commerce for Windows NT Installing and Getting Started Version 3.1*, GC09-2626
- *Net.Commerce for AIX Installing and Getting Started*, GC09-2627
- *WebSphere Application Server for OS/390 HTTP Server Planning, Installing, and Using,* SC31-8690
- *Porting a UNIX Application to OS/390 UNIX - Problems Encountered and Lessons Learned* (IBM Whitepaper available at: `http://www.s390.ibm.com/oe/aixdb2/aixdb2.html`)

B.4 Product documentation

The following is product documentation and can only be acquired through purchase of the product:

- *Program Directory for Net.Data for OS/390 with National Language Features*, GI10-6971

B.5 Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://www.ibm.com/support/techdocs/`
  IBM technical support home page for hints and tips, frequently asked questions, flashes, white papers, presentations, and tools.
- `http:www.software.ibm.com/data/net.data/library.html/`
  IBM online Net.Data manual library in html and pdf formats.
- `http://www.s390.ibm.com/oe/bpxa1por.html/`
  UNIX System Service porting page.
- `http://www.s390.ibm.com/ftp/os390/oe/docs/port.pdf/`
  Download of the *UNIX System Services Porting Guide* in pdf format.

- `http://www.s390.ibm.com/ecommerce/`
  e.commerce home page with link to Net.Commerce online documentation.
- `http://www.s390.ibm.com/products/oe/bpxa1p03.html`
- `http://www.s390.ibm.com/oe/bpxa1toy.html`
- `http://www.s390.ibm.com/oe/aixdb2/aixdb2.html`
- `http://www.ibm.com/software/data/net.data/library.html`
- `http://www.hostname.com/documents/today/text.html`
- `http://www.networking.ibm.com/nsg/nsgmain.h`

How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** 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 IBM 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 Redbooks fax order form

Please send me the following:

Title	**Order Number**	**Quantity**

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number

☐ Credit card number

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Glossary

ASCII (American National Standard Code for Information Interchange). The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

AIX Operating System. IBMs implementation of the UNIX operating system. The RS/6000 system, among others, runs the AIX operating system.

American National Standards Institute (ANSI). An organization consisting of producers, consumers, and general interest groups, that establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States.

Application Programming Interface (API). A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.

Central Processing Unit (CPU). The part of a computer that includes the circuits that control the interpretation and execution of instructions. A CPU is the circuitry and storage that executes instructions. Traditionally, the complete processing unit was often regarded as the CPU, whereas today, the CPU is often a microchip. In either case, the centrality of a processor or processing unit depends on the configuration of the system or network in which it is used.

Command. (1) A directive to the shell to perform a particular task. [POSIX.2] (2) A unit of input to an interactive program. The syntax of a command depends on the program processing it. A frequently used program is the shell, the input of which consists mostly of command lines.
(3) A request to perform an operation or run a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command. [OSF]
(4) A request from a terminal for the performance of an operation or execution of a program.
(5) A character string from a source external to a system that represents a request for system action.
(6) One or more words that tell the system to perform an operation or run a program. It consists of a name and, optionally, one or more operands.
(7) The typed name and parameters associated with an action that can be performed by an application. A command is one form of action request. Users type in the command and enter it. The computer performs the action requested by the command name.
(8) An instruction a user can type on the command line or in a CLIST.
(9) A common action users request to interact with the command area.

Customer Information Control System (CICS). An IBM licensed program that provides online transaction processing services and management for critical business applications. CICS runs on many IBM and non-IBM platforms (from the desktop to the mainframe) and is used in various types of networks that range in size from a few terminals to many thousands of terminals. The CICS application programming interface (API) enables programmers to port applications among the hardware and software platforms on which CICS is available. Each product in the CICS family can interface with the other products in the CICS family, thus enabling interproduct communication.

Cookie. Information that a Web server stores on a user's computer when the user browses a particular Web site. This information helps the Web server track such things as user preferences and data that the user may submit while browsing the site. For example, a cookie may include information about the purchases that the user makes (if the Web site is a shopping site). The use of cookies enables a Web site to become more interactive with its users, especially on future visits.

Database. (1) A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users.
(2) A collection of interrelated data organized according to a database schema to serve one or more applications.
(3) A collection of data fundamental to a system.
(4) A collection of data fundamental to an enterprise.

Database Administrator (DBA). A person who is responsible for a database system, particularly for defining the rules by which data is stored and accessed. Usually, the database administrator is also responsible for database integrity, security, performance, and recovery.

DB2. An IBM relational database management system that is available as a licensed program on several operating systems. Programmers and users of DB2 can create, access, modify, and delete data in relational tables using a variety of interfaces.

Direct Access Storage Device (DASD). A mass storage medium on which a computer stores data. Contrast with random access memory.

Dynamic Link Library (DLL). A file containing executable code and data bound to a program at load time or run time, rather than during linking. The code and data in a dynamic link library can be shared by several applications simultaneously.

Ethernet. A 10-Mbps baseband local area network that allows multiple stations to access the transmission medium at will without prior coordination, avoids contention by using carrier sense and deference, and resolves contention by using collision detection and delayed retransmission. Ethernet uses carrier sense multiple access with collision detection (CSMA/CD).

Extended Binary-coded Decimal Interchange Code (EBCDIC). A coded character set of 256 8-bit characters.

e-business. Either (a) the transaction of business over an electronic medium, such as the Internet or (b) any organization (for example, commercial, industrial, nonprofit, educational, or governmental) that transacts its business over an electronic medium, such as the Internet. An e-business combines the resources of traditional information systems with the vast reach of an electronic medium, such as the Internet (including the World Wide Web, intranets, and extranets); it connects critical business systems directly to critical business constituencies--customers, employees, and suppliers. The key to becoming an e-business is building a transaction-based Web site in which all core business processes (especially all processes that require a dynamic and interactive flow of information) are put online to improve service, cut costs, and sell products.

File Transfer. The transfer of one or more files from one system to another over a data link.

File Transfer Protocol (FTP). In the Internet suite of protocols, an application layer protocol that uses TCP and Telnet services to transfer bulk-data files between machines or hosts.

Firewall. (1) In communication, a functional unit that protects and controls the connection of one network to other networks. The firewall (a) prevents unwanted or unauthorized communication traffic from entering the protected network and (b) allows only selected communication traffic to leave the protected network.
(2) In equipment, a partition used to control the spread of fire.

Frame. (1) In Open Systems Interconnection architecture, a data structure pertaining to a particular area of knowledge and consisting of slots that can accept the values of specific attributes and from which inferences can be drawn by appropriate procedural attachments.
(2) The unit of transmission in some local area networks, including the IBM Token-Ring Network. It includes delimiters, control characters,

information, and checking characters.
(3) In SDLC, the vehicle for every command, every response, and all information that is transmitted using SDLC procedures.

Gigabyte (GB). (1) For processor storage, real and virtual storage, and channel volume, 230 or 1 073 741 824 bytes.
(2) For disk storage capacity and communications volume, 1,000,000,000 bytes.

Graphical Interchange Format (GIF). A digital format that is used to compress and transfer graphical information over computer networks. For example, GIF is a common format for graphical information on the Internet.

Home Page. The initial Web page that is returned by a Web site when a user specifies the uniform resource locator (URL) for the Web site. For example, if a user specifies the URL for the IBM Web site, which is http://www.ibm.com, the Web page that is returned is the IBM home page. Essentially, the home page is the entry point for accessing the contents of the Web site. The home page may sometimes be called the "welcome page" or the "front page."

Host. (1) A computer that is connected to a network (such as the Internet or an SNA network) and provides an access point to that network. Also, depending on the environment, the host may provide centralized control of the network. The host can be a client, a server, or both a client and a server simultaneously.
(2) In a Tivoli environment, a computer that serves as a managed node for a profile distribution.
(3) See host processor.
(4) To provide the software and services for managing a Web site.

Host Address. See IP address.

Host Processor. (1) A processor that controls all or part of a user application network.
(2) In a network, the processing unit in which the data communication access method resides.

Hypertext Markup Language (HTML). A markup language that conforms to the SGML standard and was designed primarily to support the online display of textual and graphical information that includes hypertext links.

Hypertext Transfer Protocol (HTTP). In the Internet suite of protocols, the protocol that is used to transfer and display hypertext documents.

ID. (1) Identification.
(2) Identifier.

Interactive System Productivity Facility (ISPF). An IBM licensed program that serves as a full-screen editor and dialogue manager. Used for writing application programs, it provides a means of generating standard screen panels and interactive dialogues between the application programmer and terminal user.

Internet. The worldwide collection of interconnected networks that use the Internet suite of protocols and permit public access.

Internet Architecture Board (IAB). The technical body that oversees (at a high level) the work of the Internet Engineering Task Force (IETF). The IAB approves the membership of the IETF.

Internet Engineering Task Force (IETF). The task force of the Internet Architecture Board (IAB) that is responsible for solving the short-term engineering needs of the Internet. The IETF consists of numerous working groups, each focused on a particular problem. Internet standards are typically developed or reviewed by individual working groups before they can become standards.

Internet Protocol (IP). In the Internet suite of protocols, a connectionless protocol that routes data through a network or interconnected networks and acts as an intermediary between the higher protocol layers and the physical network.

Internet Suite Of Protocols. A set of protocols developed for use on the Internet and published as Requests for Comments (RFCs) through the Internet Engineering Task Force (IETF).

IP Address. The unique 32-bit address that specifies the location of each device or workstation on the Internet. For example, 9.67.97.103 is an IP address.

Java. An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

Java Database Connectivity (JDBC). An application programming interface (API) that has the same characteristics as Open Database Connectivity (ODBC) but is specifically designed for use by Java database applications. Also, for databases that do not have a JDBC driver, JDBC includes a JDBC to ODBC bridge, which is a mechanism for converting JDBC to ODBC; it presents the JDBC API to Java database applications and converts this to ODBC. JDBC was developed by Sun Microsystems, Inc. and various partners and vendors.

Java Development Kit (JDK). A software package that can be used to write, compile, debug, and run Java applets and applications.

Java Script. A scripting language that resembles Java and was developed by Netscape for use with the Netscape browser.

Java Runtime Environment (JRE). A subset of the Java Development Kit (JDK) that contains the core executables and files that constitute the standard Java platform. The JRE includes the Java Virtual Machine, core classes, and supporting files.

Java Virtual Machine (JVM). A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

JPEG. A standard format for storing compressed true-color images. "JPEG" represents "Joint Photographic Experts Group," which is the name of the committee that developed this standard format.

Local Area Network (LANt). (1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation.
(2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network.
(3) See Ethernet and token ring.
(4) Contrast with metropolitan area network and wide area network.

Mainframe. A computer, usually in a computer center, with extensive capabilities and resources to which other computers may be connected so that they can share facilities.

Megabyte (MB). (1) For processor storage, real and virtual storage and channel volume, 220 or 1 048 576 bytes.
(2) For disk storage capacity and communications volume, 1 000 000 bytes.

Memory. All of the addressable storage space in a processing unit and other internal storages that is used to execute instructions.

Metropolitan Area Network (MAN). A network formed by the interconnection of two or more networks that may operate at higher speed than those networks, may cross administrative boundaries, and may use multiple access methods. Contrast with local area network and wide area network.

Multiple Virtual Storage (MVS). Implies MVS/390, MVS/XA, MVS/ESA, and the MVS element of the OS/390 operating system.

Network. (1) An arrangement of nodes and connecting branches.
(2) A configuration of data processing devices and software connected for information interchange.
(3) A group of nodes and the links interconnecting them.

Open Database Connectivity (ODBC). A standard application programming interface (API) for accessing data in both relational and non-relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface.

ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group and was developed by Digital Equipment Corporation (DEC), Lotus, Microsoft, and Sybase. Contrast with Java Database Connectivity.

OpenEdition. Pertaining to the elements of OS/390 that incorporate the UNIX interfaces standardized in POSIX.

OS/390. Pertaining to the IBM operating system that includes and integrates functions previously provided by many IBM software products (including the MVS operating system) and (a) is an open, secure operating system for the IBM S/390 family of enterprise servers, (b) complies with industry standards, (c) is Year 2000 ready and enabled for network computing and e-business, and (d) supports technology advances in networking server capability, parallel processing, and object-oriented programming.

Partitioned Data Set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

Personal Computer (PC). (1) A microcomputer primarily intended for stand-alone use by an individual. (2) A desktop, floor-standing, or portable microcomputer that usually consists of a system unit, a display monitor, a keyboard, one or more diskette drives, internal fixed-disk storage, and an optional printer. PCs are designed primarily for stand-alone operation but may be connected to mainframes or networks.

Program Temporary Fix (PTF). A temporary solution or bypass of a problem diagnosed by IBM in a current unaltered release of the program.

Random Access Memory (RAM). A temporary storage location in which the central processing unit (CPU) stores and executes its processes. Contrast with direct access storage device.

Resource Access Control Facility (RACF). An IBM licensed program that provides access control by identifying users to the system, verifying users of the system, authorizing access to protected resources, logging detected, unauthorized attempts to enter the system; and logging detected accesses to protected resources. RACF is included in OS/390 Security Server and is also available as a separate program for the MVS and VM environments.

RS/6000. A family of workstations and servers based on IBM's POWER architecture. They are primarily designed for running multi-user numerical computing applications that use the AIX operating system.

Secure Sockets Layer (SSL). A security protocol that provides communication privacy. SSL enables client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc.

SQL. A programming language that is used to define and manipulate data in a relational database.

System Modification Program Extended (SMP/E). An IBM licensed program used to install software and software changes on MVS systems. In addition to providing the services of SMP, SMP/E consolidates installation data, allows more flexibility in selecting changes to be installed, provides a dialog interface, and supports dynamic allocation of data sets.

S/390. Pertaining to the family of IBM enterprise servers that demonstrate outstanding reliability, availability, scalability, security, and capacity in today's network computing environments.

Time Sharing Option (TSO). An option of the MVS operating system that provides interactive time sharing from remote terminals.

Telnet. In the Internet suite of protocols, a protocol that provides remote terminal connection service. It allows users of one host to log on to a remote host and interact as directly attached terminal users of that host.

Token Ring. (1) According to IEEE 802.5, network technology that controls media access by passing a token (special packet or frame) between media-attached stations.

(2) A FDDI or IEEE 802.5 network with a ring topology that passes tokens from one attaching ring station (node) to another.
(3) See local area network.

Transmission Control Protocol (TCP). A communications protocol used in the Internet and in any network that follows the Internet Engineering Task Force (IETF) standards for internetwork protocol. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It uses the Internet Protocol (IP) as the underlying protocol.

Transmission Control Protocol/Internet Protocol (TCP/IP). (1) The Transmission Control Protocol and the Internet Protocol, which together provide reliable end-to-end connections between applications over interconnected networks of different types.
(2) The suite of transport and application protocols that run over the Internet Protocol.

Uniform Resource Locator (URL). (1) A sequence of characters that represent information resources on a computer or in a network such as the Internet. This sequence of characters includes (a) the abbreviated name of the protocol used to access the information resource and (b) the information used by the protocol to locate the information resource. For example, in the context of the Internet, these are abbreviated names of some protocols used to access various information resources: http, ftp, gopher, telnet, and news; and this is the URL for the IBM home page: `http://www.ibm.com`
(2) The address of an item on the World Wide Web. It includes the protocol followed by the fully qualified domain name (sometimes called the host name) and the request. The Web server typically maps the request portion of the URL to a path and file name. For example, if the URL is `http://www.networking.ibm.com/nsg/nsgmain.htm`, the protocol is http; the fully qualified domain name is `www.networking.ibm.com`; and the request is `/nsg/nsgmain.htm`.

UNIX Operating System. An operating system developed by Bell Laboratories that features multi-programming in a multi-user environment. The UNIX operating system was originally developed for use on minicomputers but has been adapted for mainframes and microcomputers. The AIX operating system is IBMs implementation of the UNIX operating system.

Upload. (1) To transfer programs or data from a connected device, typically a personal computer, to a computer with greater resources.
(2) To transfer data from a device, such as a workstation or a microcomputer, to a computer.

Web Browser. A client program that initiates requests to a Web server and displays the information that the server returns.

Web Page. Any document that can be accessed by a uniform resource locator (URL) on the World Wide Web. Contrast with home page.

Web Server. A server that is connected to the Internet and is dedicated to serving Web pages.

Web Site. A Web server that is managed by a single entity (an organization or an individual) and contains information in hypertext for its users, often including hypertext links to other Web sites. Each Web site has a home page. In a uniform resource locator (URL), the Web site is indicated by the fully qualified domain name. For example, in the URL `http://www.networking.ibm.com/nsg/nsgmain.htm`, the Web site is indicated by `www.networking.ibm.com`, which is the fully qualified domain name.

WebSphere. Pertaining to a family of IBM software products that provide a development and deployment environment for basic Web publishing and for transaction-intensive, enterprise-scale e-business applications.

Wide Area Network (WAN). (1) A network that provides communication services to a geographic area larger than that served by a local area network or a metropolitan area network, and that may use or provide public communication facilities.
(2) A data communication network designed to serve an area of hundreds or thousands of miles; for example, public and private packet-switching

networks, and national telephone networks.
(3) Contrast with local area network and metropolitan area network.

Working Directory. The directory that is currently in use by an operating system or application. If no path is specified, this is the directory to which data is written, from which data is deleted, or in which data is searched.

Workstation. (1) A functional unit at which a user works. A workstation often has some processing capability.
(2) One or more programmable or non-programmable devices that allow a user to do work.
(3) A terminal or microcomputer, usually one that is connected to a mainframe or to a network, at which a user can perform applications.

World Wide Web (WWW). A network of servers that contain programs and files. Many of the files contain hypertext links to other documents available through the network.

Virtual Machine (VM). (1) A virtual data processing system that appears to be at the exclusive disposal of a particular user, but whose functions are accomplished by sharing the resources of a real data processing system.
(2) In VM/ESA, the virtual processors, virtual storage, virtual devices, and virtual channel subsystem allocated to a single user. A virtual machine also includes any expanded storage dedicated to it.
(3) See Java Virtual Machine.

Abbreviations and acronyms

AIX	AIX Operating System
ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American National Standard Code for Information Interchange
CICS	Customer Information Control System
CLI	Command Line Interface
CPU	Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DASD	Direct Access Storage Device
DBA	Database Administrator
DLL	Dynamic Link Library
EBCDIC	Extended Binary-Coded Decimal Interchange Code
FTP	File Transfer Protocol
GB	Gigabyte
GIF	Graphical Interchange Format
GUI	Graphical User Interface
HFS	Hierarchical File System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAB	Internet Architecture Board
IBM	International Business Machines Corporation
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPC	Inter-Process Communication
ISPF	Interactive System Productivity Facility
ITSO	International Technical Support Organization
JDBC	Java Database Connectivity
JDK	Java Development Kit
JPEG	Joint Photographic Experts Group
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MB	Megabyte
MVS	Multiple Virtual Storage
MVS/ESA	Multiple Virtual Storage/Extended Storage Architecture
MVS/XA	Multiple Virtual Storage/Extended Architecture
NLS	National Language Support
ODBC	Open Database Connectivity
PDS	Partitioned Data Set
PC	Personal Computer
PTF	Program Temporary Fix

RACF	Resource Access Control Facility
RAM	Random Access Memory
RFC	Requests for Comments
SSL	Secure Sockets Layer
SMP/E	System Modification Program Extended
SPFUI	SQL Processor Using File Input
SQL	Structured Query Language
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TSO	Time Sharing Option
UDB	Universal Database
URL	Uniform Resource Locator
WWW	World Wide Web
VM	Virtual Machine

Index

IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at `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`

Document Number **Redbook Title**	SG24-5516-00 Net.Commerce: Develop on NT or AIX, Deploy on S/390
Review	
What other subjects would you like to see IBM Redbooks address?	
Please rate your overall satisfaction:	O Very Good O Good O Average O Poor
Please identify yourself as belonging to one of the following groups:	O Customer O Business Partner O Solution Developer O IBM, Lotus or Tivoli Employee O None of the above
Your email address: The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	O Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
Questions about IBM's privacy policy?	The following link explains how we protect your personal information. `http://www.ibm.com/privacy/yourprivacy/`

SG24-5516-00
Printed in the U.S.A.

Net.Commerce: Develop on NT or AIX, Deploy on S/390

SG24-5516-00