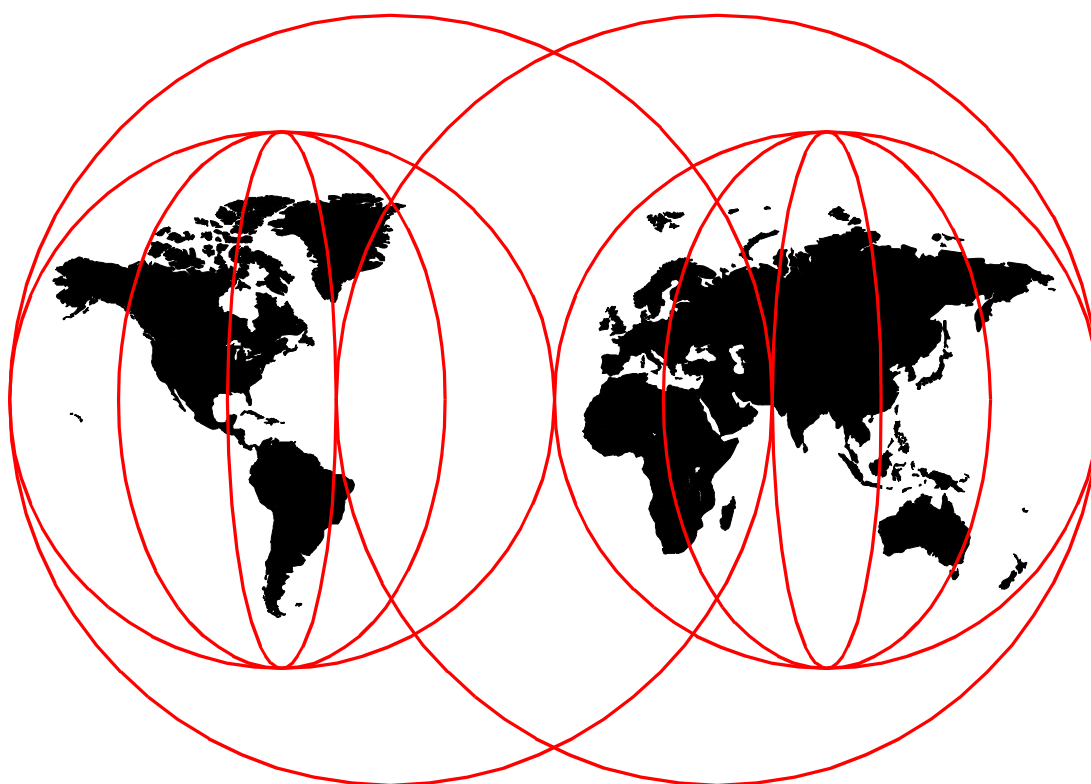


Migrating Net.Commerce Applications to OS/390

*Erich Amrehn, Daniel Bourque, Nick Carlin,
David Hauser, Vinod V. Kumar*



International Technical Support Organization

www.redbooks.ibm.com



International Technical Support Organization

SG24-5438-00

**Migrating Net.Commerce Applications
to OS/390**

March 2000

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, "Special notices" on page 111.

First Edition (March 2000)

This edition applies to Net.Commerce for OS/390 Version 3 Release 1.2, Program Number 5697-D32, and DB2 for OS/390 Version 5, Program Number 5655-DB2, for use with the OS/390 operating system.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

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

© Copyright International Business Machines Corporation 2000. All rights reserved.

Note to U.S Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	vii
Preface	ix
The team that wrote this redbook	ix
Comments welcome	x
Chapter 1. Net.Commerce for OS/390	1
1.1 Who is this book for?	1
1.2 Documentation on the Web	1
1.3 IBM Education classes	1
1.4 Review of Net.Commerce	2
1.5 Project management considerations	3
1.5.1 Skills	4
1.5.2 Incremental testing	5
1.5.3 Solutions Assurance	5
1.5.4 Moving to production on OS/390	6
1.5.5 Sample project plan	7
1.5.6 IBM e-business mark and security testing	7
Chapter 2. Platform differences	9
2.1 When is the S/390 platform a good choice?	9
2.2 Relative platform sizing	11
2.3 Differences between solution architectures	12
2.3.1 Differences in staging environments	15
2.3.2 Windows NT and UNIX testing environment	15
2.3.3 OS/390 testing environment	16
Chapter 3. General migration considerations	19
3.1 ASCII to EBCDIC	19
3.1.1 Typical problem areas	19
3.1.2 File transfer using TCP/IP FTP	20
3.1.3 Net.Commerce issues	20
3.1.4 Shopper password porting	21
3.2 Net.Data compatibility	21
3.2.1 Path names in Net.Data macros	22
3.2.2 DTW_ODBC must be changed to DTW_SQL	22
3.2.3 Using the UNIX sed command for global changes	22
3.2.4 Net.Data special characters	23
3.2.5 Net.Data functional comparison	23
3.3 HTML and image considerations	24
3.4 Product Advisor compatibility	25
3.5 Catalog Architect compatibility	25
3.6 Payment Server	26
3.7 Backend integration	26
3.7.1 CICS integration	26
3.7.2 IMS integration	27
3.7.3 Commerce Integrator	27
3.8 Common Connector Framework	28
3.9 Commands, Tasks, and Overrideable Functions	28
3.9.1 Commands and Overrideable Functions	29
3.9.2 Tasks	31

3.10	Installation and operation differences between platforms	32
3.10.1	AIX installation overview	33
3.10.2	Windows NT installation overview	33
3.10.3	OS/390 installation overview	34
3.11	Configuration	35
3.11.1	UNIX and Windows NT Net.Commerce configuration	35
3.11.2	OS/390 Net.Commerce configuration	36
3.11.3	Input to CMNCONF	37
3.11.4	Output from CMNCONF	38
3.11.5	Placing your HTML files in a non-default directory	40
3.11.6	Placing your Net.Data macros in a non-default directory	41
3.11.7	Placing Net.Commerce configuration files into a single directory	42
3.11.8	Installation directories	45
3.12	Net.Commerce operation	45
3.12.1	AIX operation	45
3.12.2	Windows NT operation	45
3.12.3	OS/390 operation	45
3.13	Maintaining an OS/390 Net.Commerce site	47
3.14	Summary of the general migration considerations	47
Chapter 4.	Migrating the database	49
4.1	Overview	49
4.1.1	Determine the database objects the user has added/changed	51
4.1.2	Differences between NT and OS/390 Net.Commerce tables	56
4.1.3	Common tables of Windows NT and OS/390 Net.Commerce	62
4.1.4	Determine high-level DB2 for OS/390 database design	64
4.1.5	Data definition language	65
4.1.6	Methods to migrate data	68
4.1.7	Database housekeeping	73
4.2	Summary	75
Chapter 5.	Application development	77
5.1	Microsoft Visual C++ development	77
5.2	UNIX and OS/390 command line development	78
5.3	OS/390 C++ compilation tools	80
5.3.1	Example of an OS/390 makefile	80
5.4	DB2 application development on OS/390	81
5.4.1	SQL from C with Net.Commerce V3.1 and V3.1.1	81
Appendix A.	Sample customer scenario for migration and porting	83
A.1	Copying a Windows NT Net.Commerce database	83
Appendix B.	Solutions Assurance and stress testing	85
B.1	Solutions Assurance	85
B.2	Stress testing	85
B.2.1	Stress test	85
B.2.2	Load test	86
B.3	Pretest strategy	86
B.4	Script development	87
B.5	Outline of test runs	87
B.6	Stress test environment	88
Appendix C.	Net.Commerce platform function comparison	89
C.1	Differences between system architectures	91

C.1.1 AIX features	92
C.1.2 Windows NT features	93
C.1.3 OS/390 features	94
Appendix D. OS/390 Net.Commerce development tools	97
D.1 OS/390 Net.Commerce sample makefile	97
D.2 Precompiling C code containing static SQL with Net.Commerce V3.1.1	99
Appendix E. REXX sample to generate DB2 for OS/390 utility job.	105
E.1 Reorganize and copy REXX sample.	105
E.2 RUNSTATS REXX sample	105
Appendix F. Alternative approach to database migration	107
F.1 DB2 housekeeping	109
Appendix G. Special notices	111
Appendix H. Related publications	115
H.1 IBM Redbooks publications.	115
H.2 IBM Redbooks collections.	115
H.3 Other resources	115
H.4 Referenced Web sites.	116
How to get IBM Redbooks	117
IBM Redbooks fax order form.	118
Glossary	119
Index	121
IBM Redbooks evaluation	127

Figures

1. Overview of the Net.Commerce architecture	3
2. Skills required for a successful Net.Commerce migration project	5
3. Qualitative sizing of Net.Commerce solutions across different platforms	12
4. Typical Windows NT or UNIX Net.Commerce system	13
5. Typical OS/390 Net.Commerce system	14
6. Possible OS/390 Net.Commerce configuration	15
7. Windows NT and UNIX staging environment	16
8. OS/390 testing environment	17
9. Porting Net.Data macros from Windows NT to OS/390	22
10. HTML and image porting	24
11. Connecting to backend CICS	27
12. Overview of Commands, Tasks, and Overrideable Functions	29
13. Migration of Net.Commerce Commands and Overrideable Functions	30
14. A high-level overview of installation and configuration on AIX	33
15. A high-level overview of installation and configuration on Windows NT	34
16. A high-level overview of installation and configuration on OS/390	34
17. Windows NT/UNIX Net.Commerce configuration screen	36
18. OS/390 CMNCONF screen	37
19. Output from CMNCONF	39
20. Net.Commerce HFS hierarchy	43
21. Symbolic links required for config files all go in the same directory	44
22. DB2 migration overview	50
23. DB2LOOK syntax	66
24. Translate table and index DDL	68
25. Microsoft Visual C++ 6.0 compilation of a Net.Commerce Command	78
26. rlogin session to OS/390	79
27. ISPF editor session	80
28. Generic stress test	86
29. Generic load test	86

Preface

This redbook will help you to migrate Net.Commerce solutions from distributed platforms (defined in this book as Windows NT and UNIX) to OS/390. This publication is intended to help project leaders and technical specialists who are working with Net.Commerce understand the tasks involved with the migration of Net.Commerce for Windows NT or UNIX to Net.Commerce for OS/390.

First, we introduce Net.Commerce for OS/390 and the project steps required for a migration from the distributed platform solutions. This will be especially useful for business and project managers with responsibility for the success of the migration.

Next, we highlight the differences between the OS/390 and distributed platforms. Information about general migration considerations and compatibility issues are discussed.

Migrating the central part of any Net.Commerce solution—the catalog database (DB2)—is described. The differences in developing Net.Commerce customized code on the OS/390 and distributed platforms are also discussed.

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, Poughkeepsie Center.

Erich Amrehn is a certified Senior IT Specialist at the International Technical Support Organization, Poughkeepsie Center. Before joining the ITSO, he worked as a technical consultant to the IBM System/390 division for e-commerce on S/390 in Europe, the Middle East, and Africa. He also has 13 years of VM experience in various technical positions in Germany and other areas in Europe and worldwide.

Daniel Bourque is an IMS Product Specialist based out of the Santa Teresa Software Laboratory in San Jose, CA. He holds a degree in Management/Information Science and Finance from the University of Florida. Daniel has three years experience in the IT industry and specializes in System/390 database management and connectivity.

Nick Carlin is the EMEA Beta Program Manager for Net.Commerce/390, based in Hursley Park in the United Kingdom. He has four years of experience working with customers implementing OS/390 e-business solutions. He has worked with IBM for 14 years. His areas of expertise include Net.Commerce for OS/390 and Tivoli Systems Management for OS/390. He writes both IBM and industry publications in these subjects and is a regular Guide/Share speaker.

David Hauser is an Advisory Programmer at IBM Santa Teresa Lab. He has 15 years of experience in the IBM DB2 for S/390 Development and Performance organization. Dave is now a member of the Net.Commerce for OS/390 Development team, specializing in DB2 matters.

Vinod V. Kumar has been a Technical Manager in Learning Services at IBM Global Services India since Nov 1997. He has two years of experience in

teaching MVS and OS/390 courses, three years in software development, and two years in teaching. His areas of expertise include operating systems, networks, speech processing, and multimedia. Recently, he has become involved with e-business technologies on OS/390.

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

Don Bagwell
Washington Systems Center, IBM Gaithersburg

Cynthia Bartholome
Customer Enablement, IBM Poughkeepsie

Nils Begquist
IBM Global Services, IBM Gothenburg

Rich Conway
International Technical Support Organization, Poughkeepsie Center

Kevin Curley
S/390 New Technology Center, IBM Poughkeepsie

Roland Doemer
International Technical Support Organization, San Jose Center

Dawn Hamilton
Product Introduction Programs, IBM Poughkeepsie

Jack Hoarau
EMEA S/390 New Technology Center, IBM Montpellier

Bruce McAlister
DB2 Development, IBM San Jose

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 evaluation" on page 127 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. Net.Commerce for OS/390

Net.Commerce is the premier “merchant server” software from IBM. It was originally written for AIX but today runs on AIX, Sun Solaris, Windows NT, OS/400, and OS/390.

Version 1 of Net.Commerce was first available for OS/390 in June 1997. At the time of writing this book, Net.Commerce for OS/390 is at Version 3.1.2.

1.1 Who is this book for?

We assume that your Net.Commerce Web site today is running successfully on Windows NT or UNIX and you are now ready to grow to the OS/390 solution.

We envision at least three different types of individuals who need to read this book:

- The project manager who is tasked with migrating a Net.Commerce solution from UNIX or Windows NT to OS/390.
- A UNIX or Windows NT Net.Commerce application developer who needs to appreciate the differences between OS/390 and the distributed platforms.
- An OS/390-skilled individual who is involved in providing either a system or the application development skill to host the migration of a distributed Net.Commerce application.

For reasons of when and why migration to the OS/390 platform is appropriate, see 2.1, “When is the S/390 platform a good choice?” on page 9.

1.2 Documentation on the Web

In addition to reading this book, you should also read the OS/390 Net.Commerce documentation available on the Internet at:

<http://www.s390.ibm.com/nc/ecommerce/doc.html#net>

1.3 IBM Education classes

The following IBM classes are available. Note that the course number starting with “IN” is the worldwide course reference number and the number after the “/” is the course reference number in the USA.

- IN97/N3315 – Net.Commerce V3.1.2 Implementation (labs on Windows NT)
- IN96/N3385 – Net.Commerce V3.1.2 Customization (labs on Windows NT)
- IN58/N3415 – Net.Commerce V3.2 for AS/400 Implementation
- IN59/N3485 – Net.Commerce V3.2 for AS/400 Customization
- IN98/N3390 – Net.Commerce Implementation for OS/390
- IN55/N3395 – Net.Commerce Hosting Server V3.1.1
- IN95/N3305 – Net.Commerce Design and Integration
- IN45/D3300 – Catalog Architect (Web book)

- IN57 – Payment Server (Web book)

Contact your IBM representative for more information.

1.4 Review of Net.Commerce

This section should be a review for many people already familiar with Net.Commerce.

Users access the OS/390 Net.Commerce system through HTTP requests to a Web server. The Web server examines the incoming URL, and if the URL contains a Net.Commerce Command, passes the Command string to the Net.Commerce director, which is a Web server plug-in that runs in the Web server address space.

The Net.Commerce director communicates over a local IP socket to the main Net.Commerce daemon, which runs in a different address space to the Web server. The main Net.Commerce daemon has a number of child processes running in its address space that communicate to the various backend systems.

For product and control information, Net.Commerce itself uses a DB2 database that is generally accessed through Net.Data, and Net.Commerce Commands and Overrideable Functions, which are written in C++.

Net.Data is used as the main display engine for the Net.Commerce system. As the display engine, Net.Data should not be used to implement business logic. The actual piece of code that is run and generates HTML is called a *macro*. The Net.Data product that is shipped with Net.Commerce is actually a special version of Net.Data built to integrate directly in Net.Commerce. Net.Data does not have to be separately installed or configured. The Net.Data that is shipped with Net.Commerce will coexist with a stand-alone installed copy of Net.Data.

The team creating the site can implement their own Net.Data macros, Commands, and Overrideable Functions. The ability to do this gives Net.Commerce great flexibility and makes it a marketplace winner. Net.Commerce was recently awarded first place in a comparison against the other major merchant servers available in the marketplace today ("Which Commerce Platform?" October 1999 report from Forrester Research Inc.).

Figure 1 on page 3 shows an overview of the Net.Commerce architecture.

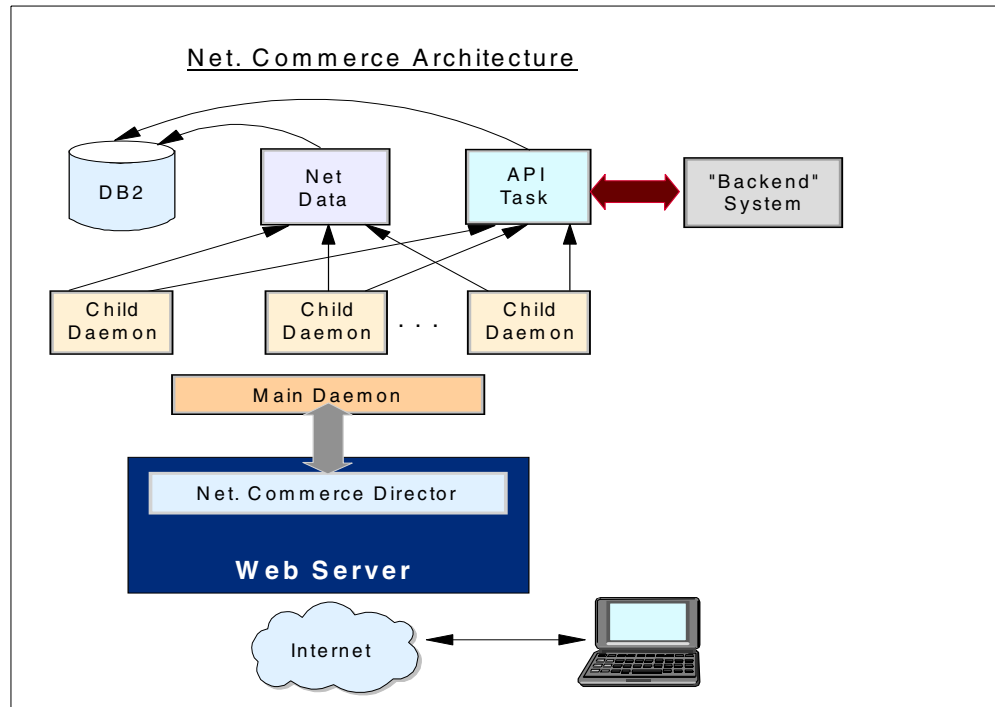


Figure 1. Overview of the Net.Commerce architecture

OS/390 Net.Data is shipped with OS/390 Net.Commerce.

Many people form the impression that Net.Commerce is a “load-and-go” application that requires a few clicks and an online store is ready to go. That’s only true if you’re willing to accept the most basic store, with very basic functionality. Net.Commerce is designed to be a toolkit. It is designed to be customized to fit the needs of each customer individually. It is important to understand early that Net.Commerce will require customization, and that the projects can get very complex, depending on the nature of that customization.

As a general rule, the fewer customizations to the basic mall structure will mean the fewer porting activities that need to be completed when moving your site from UNIX or Windows NT to OS/390. OS/390 only uses the Domino Go Web server as the HTTP server (there is no Netscape Enterprise server or Apache server for OS/390 at the time of writing). Porting from the Domino Go Web server on other platforms is likely to be less of an effort. Finally, DB2 is the only database that Net.Commerce currently works with on OS/390, so porting from a DB2 environment on UNIX or Windows NT will be more straightforward.

1.5 Project management considerations

We’ve seen some Net.Commerce installation and migration projects get into difficulties. Here, we show you some of the factors to consider to achieve success and to avoid the common pitfalls.

The first consideration is to make sure that the business managers in the organization support the activity.

1.5.1 Skills

Net.Commerce is unique in that the skills required to successfully implement the solution fall outside as well as inside the technical arena. In fact, without the close input of the “business” side as well as the IT side within the organization, the project cannot succeed to any significant degree.

The application development and solution group that designed and implemented the original UNIX or Windows NT solution should be involved if possible. This is the core group. You then need to wrap these skills with the OS/390 knowledge necessary for a migration to OS/390. These skills will almost always come from *different* individuals. The main OS/390 skills required will be in:

- Base OS/390 Skills
- DB2 Systems Administration
- OS/390 Web Server
- OS/390 UNIX System Services (USS)
- OS/390 Security Server
- OS/390 Performance Tuning

This may not be a complete list; depending on the backend systems that your Net.Commerce server needs to connect to, you may also need CICS, IMS, MQSeries, or other skills.

The diagram shown in Figure 2 on page 5 is a schematic of the skills required and how they fit together.

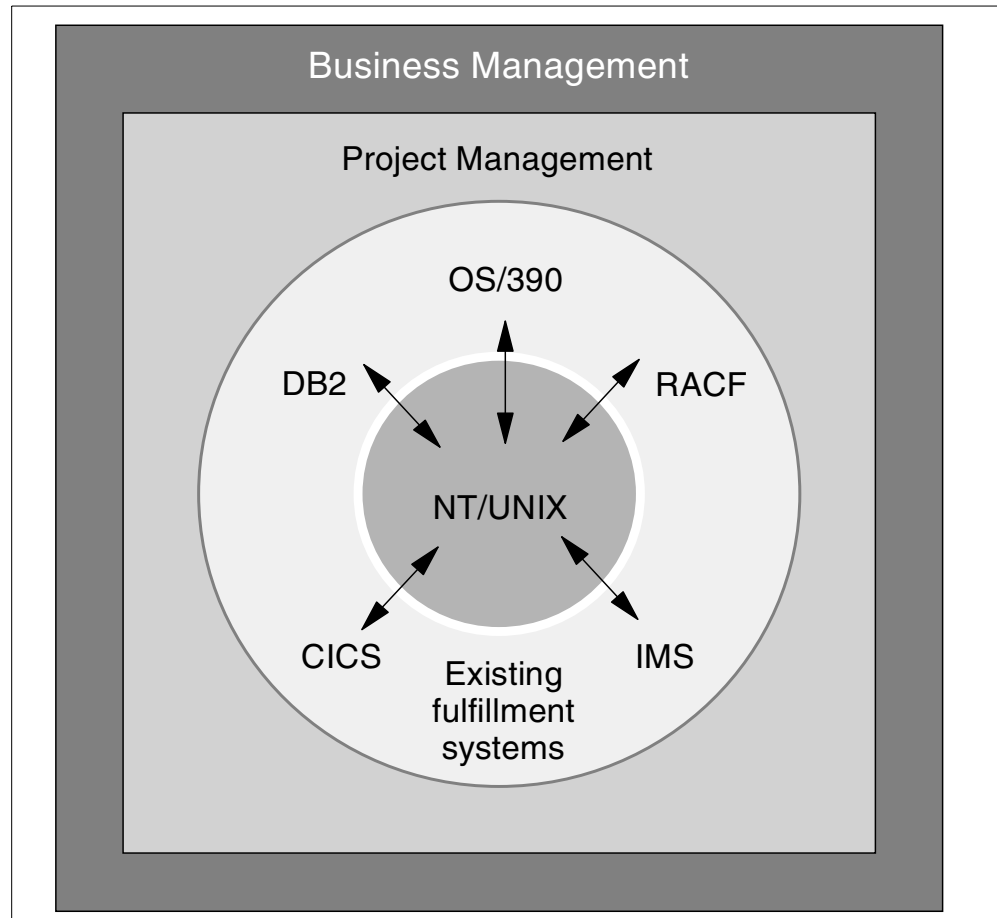


Figure 2. Skills required for a successful Net.Commerce migration project

1.5.2 Incremental testing

One approach to migration could be to port everything to OS/390 and see if it works. This is not generally recommended and testing should be carried out at each stage of the migration from UNIX or Windows NT to OS/390. This ensures that each component is verified as it is ported.

1.5.3 Solutions Assurance

Net.Commerce for OS/390 is designated for Solutions Assurance. This means that IBM recommends that all OS/390 Net.Commerce solutions go through the Solutions Assurance process to be stress tested. Contact your IBM representative for further information on Solutions Assurance.

1.5.3.1 Stress testing

As part of the Solutions Assurance process a stress test should be performed. IBM Global Services can provide a stress testing service for OS/390 Net.Commerce customers.

For a good stress test, an intranet token-ring LAN is not adequate because you want to stress the system, not the connection; you do not want any other users on that network to corrupt the performance results.

Token ring has a bandwidth of 16 Mb/sec whereas Fast Ethernet has a bandwidth of 100 Mb/sec. For a stress test of Net.Commerce solutions, IBM Global Services use a Private Fast Ethernet connection from their stress test (injector) system to their OS/390 server. This will create an isolated test bed that will lower the chance of a network bottleneck.

Before a stress test, some information should be collected by IBM from the customer intending to run a Net.Commerce site on OS/390, such as throughput objectives (expressed in users, hits, and transactions per unit of time), the percentage of shopper buying activity versus browsing expected for their Net.Commerce solution, and some description of the workload to be tested. Buying activity is usually more expensive in resource terms, due to files and databases being updated and SSL encryption costs.

For example, customers have provided a typical user scenario, or a percentage for buying versus browsing, so that stress test workload scripts can be built to emulate the correct environment. If this information is not available, IBM assumes default scenarios to measure such as:

- 95 percent browsing 5 percent buying
- 80 percent browsing 20 percent buying
- 50 percent browsing 50 percent buying.

If the customer has no knowledge of how the shopper will shop, all three scenarios should be measured.

Finally, another piece of key information needed from the customer is the expected growth of traffic on the site. It is important to use this information for capacity planning. Growth estimates can be difficult to obtain, especially in the fast and unpredictable environment of today's Internet. For more information about stress testing, see Appendix B, "Solutions Assurance and stress testing" on page 85.

1.5.4 Moving to production on OS/390

The recommended approach during a migration is to copy your Windows NT or UNIX site to another Windows NT or UNIX site, held locally, then port this copy to OS/390.

Before going live with OS/390, bring down your production site (hopefully during a regular maintenance slot) and make the final updates required to the migrated OS/390 Net.Commerce database.

Switch the target servers in your domain name server to point customers to your OS/390 site rather than your Windows NT or UNIX site. Finally, start your site on OS/390 and then users can start connecting and shopping.

Sharing the Net.Commerce database between UNIX or Windows NT and OS/390 was not explored during the writing of this redbook. It may be possible to use a data propagation tools (such as IBM Data Propagator) to propagate changes between Windows NT, UNIX, and OS/390, but that approach was not explored with this book.

1.5.5 Sample project plan

Here, we show at a very high level the major project phases that are in a migration of an existing distributed Net.Commerce application to OS/390. This book covers the activity needed in each of the following areas. The time taken for each activity will vary according to your installation, but an overall project time of three months is a reasonable estimate.

Preparation

- OS/390 environment
 - Install OS/390, Web server, Net.Commerce, and DB2
 - Create an application development environment on your OS/390 server to allow the building of Net.Commerce customization code, which is written in C++

Porting/Testing

- HTML pages
- Images
- Net.Data macros
- Database contents
- Commands, Tasks, and Overrideable Functions

Solution Testing

- Solutions assurance and stress testing

Production

- Cut over
- Operation
- Performance analysis and tuning
- Backup, recovery, and maintenance

1.5.6 IBM e-business mark and security testing

The IBM e-business mark is a symbol that tells the world that you've moved a significant business process to the Web to bring added value to your customers, partners, or suppliers. A survey conducted in early 1998 revealed that 42 percent of the Internet users surveyed indicated they were more likely to conduct transactions on sites that included the IBM e-business mark. This research finding is a powerful testament to the perceived value that IBM technology brings to the e-business world.

In addition, IBM also provides you with a free Web Site Security Scan every quarter you host the mark on your site. The Security Scan involves the IBM Emergency Response Team working with the customer to probe the customers' commerce site for common security loopholes.

See <http://www.ibm.com/e-business/emarkinfo/> for more information about the e-business mark.

Chapter 2. Platform differences

After a customer has decided to use Net.Commerce as their commerce server (rather than a competitive product), they will then decide which platform to run Net.Commerce on. If you are reading this book, you are probably considering migrating from Windows NT or UNIX to OS/390. This chapter provides information about the differences between Net.Commerce running on different platforms and when to use OS/390.

2.1 When is the S/390 platform a good choice?

Today's open, client/server network computing environments span a vast range of system sizes, from single workstations to massively parallel clusters of hundreds of processors. Within that range, the S/390 server is often the best solution for commercial processing and e-business Web serving. We refer to S/390 in this section as both the hardware and software of the platform.

There are key questions to ask a customer who is considering which platform to run their commerce server on:

- Is your data on S/390?
- Are your fulfillment systems on S/390?
- Do you have an S/390 skill base?
- Is scalability important to you?
- Is availability important to you?
- Is security important to you?
- Is manageability important to you?
- Is flexibility important to you?
- Is cost important to you?

If the customer answers yes to four or more of the above, S/390 is a good platform for them to host their commerce server.

Data

If the data is on the S/390 platform today, it is quite expensive to move it to a distributed site just to run an online store. The activity of migrating this data has to happen on a very regular basis, usually overnight, which can cost a lot in terms of bandwidth and systems management.

Fulfillment Systems

Placing the e-commerce engine in close proximity to the fulfillment systems simplifies the design and improves the system performance. Even if the backend system is on a separate LPAR or S/390, channel connectivity can be utilized that doesn't involve the heavy protocol stacks normally needed for LAN or WAN communication.

Skills

Use the S/390 as the commerce platform when the customer has an existing S/390 skill base that he or she wishes to leverage or to grow.

Scalability

S/390 can scale beyond any other platform. It is capable of scaling up to very high levels, either within a single machine or by coupling multiple systems together with a Parallel Sysplex. The use of WorkLoad Manager (WLM) can help the e-commerce site meet its service goals by shifting resources around as mixed workloads are placed on the system.

Availability

The e-commerce site will be expected to run 365 days a year and 24 hours a day. Unplanned outages can be very expensive; if the site is unavailable for any reason, the shopper may click on the link of a competitor, possibly never to return. S/390 has the highest continuous availability matrix of any commercial computer system available today. If you require continuous availability, you will require multiple Net.Commerce instances running, sharing a common database. Traffic to the site can be balanced across the multiple instances of Net.Commerce using some IP traffic balancer (IBM Net.Dispatcher, for example). That would allow you to make a change to one instance's configuration file and then stop and restart that instance. During this process, the other instances can continue servicing the traffic (provided the resources for the remaining instances is sufficient to handle the load). When the modified instance comes back online, it may then step in and continue accepting traffic. This is an example of where a Sysplex configuration would be ideal.

Performance

S/390 is very efficient in its use of hardware resources, thus minimizing the amount of hardware you need to buy. The excellent response time and throughput of the S/390 Net.Commerce system is a powerful reason to move to the platform.

Security

Security is one of the key issues facing Internet commerce today. The S/390 platform is well proven, and OS/390 UNIX System Services, using the OS/390 Security Server (RACF), helps security. We claim that OS/390 has the most secure UNIX implementation in the marketplace today.

Manageability

S/390 was designed with manageability in mind. Throughout its 30 plus years of life, OS/390 and its predecessors have focused on making the system efficient to manage. In addition, one large server is easier and more cost effective to manage than lots of small ones.

Flexibility

S/390 runs most of the commercial processing in the world today, but it also supports many of the latest interfaces and applications. It allows you to integrate the old with the new, thus providing the best in application flexibility.

Costs

Surveys consistently show that S/390 is the lowest cost solution for large commercial computing systems. This has been helped by the transition to CMOS hardware and new software pricing options, but is also due to the economies of scale from a large server and the fact that S/390 was designed for low system management costs.

Table 1 shows the main differences between the Windows NT, S/390, and UNIX platforms. Although there are some features that appear in all three columns, differences in implementation can be significant. UNIX and Windows NT do not have as high a US Department of Defense security rating as OS/390. In the table cells, a **Y** indicates that the feature or function is available for that platform, while *y* indicates reduced capability relative to the S/390 platform running OS/390.

Table 1. Business value of platform choices

Feature	S/390	UNIX	NT	Business Benefit
SMP	Y	Y	<i>y</i>	Some scale/growth
Security	Y	<i>y</i>	<i>y</i>	Security of business information
RAS	Y	<i>y</i>	<i>y</i>	Availability of system
Multiple workloads	Y	<i>y</i>	<i>y</i>	Economies of scale
Upgradeable	Y	<i>y</i>	<i>y</i>	Economy
Clustering	Y	Y	<i>y</i>	Scale/growth
Online backup	Y	<i>y</i>		Availability of system
Online reorganization	Y			Availability of system
High scalability	Y	<i>y</i>		Business will not outgrow IT system
Workload balancing	Y			Higher throughput
Automation	Y	<i>y</i>	<i>y</i>	Reduce people costs
Job scheduling	Y			Greater throughput
Management	Y	<i>y</i>	<i>y</i>	Reduce people costs
Multiple networks	Y	Y	Y	No networking constraints
Partitioning	Y	<i>y</i>		Better use of server capacity
Online reconfiguration	Y			Availability of system
Disk mirroring	Y	Y	Y	Security of business data
Run UNIX applications	Y	Y		Application choice
Run Windows NT applications	Y	Y	Y	Application choice
Web serving	Y	Y	Y	Information access
Java enabled	Y	Y	Y	Platform independence
Object oriented	Y	Y	Y	Faster application development

2.2 Relative platform sizing

Performance measurement comparing the different platforms has been undertaken at IBM, but the official results have not yet been published. Windows NT solutions are usually smaller scale than UNIX or OS/390 solutions.

The chart shown in Figure 3 shows a qualitative comparison of where the different platforms support workloads today. It should be made clear that there are many UNIX platforms supporting larger workloads than some OS/390 Web servers today. This chart shows you the overall capability of single servers across the different platforms supported by Net.Commerce.

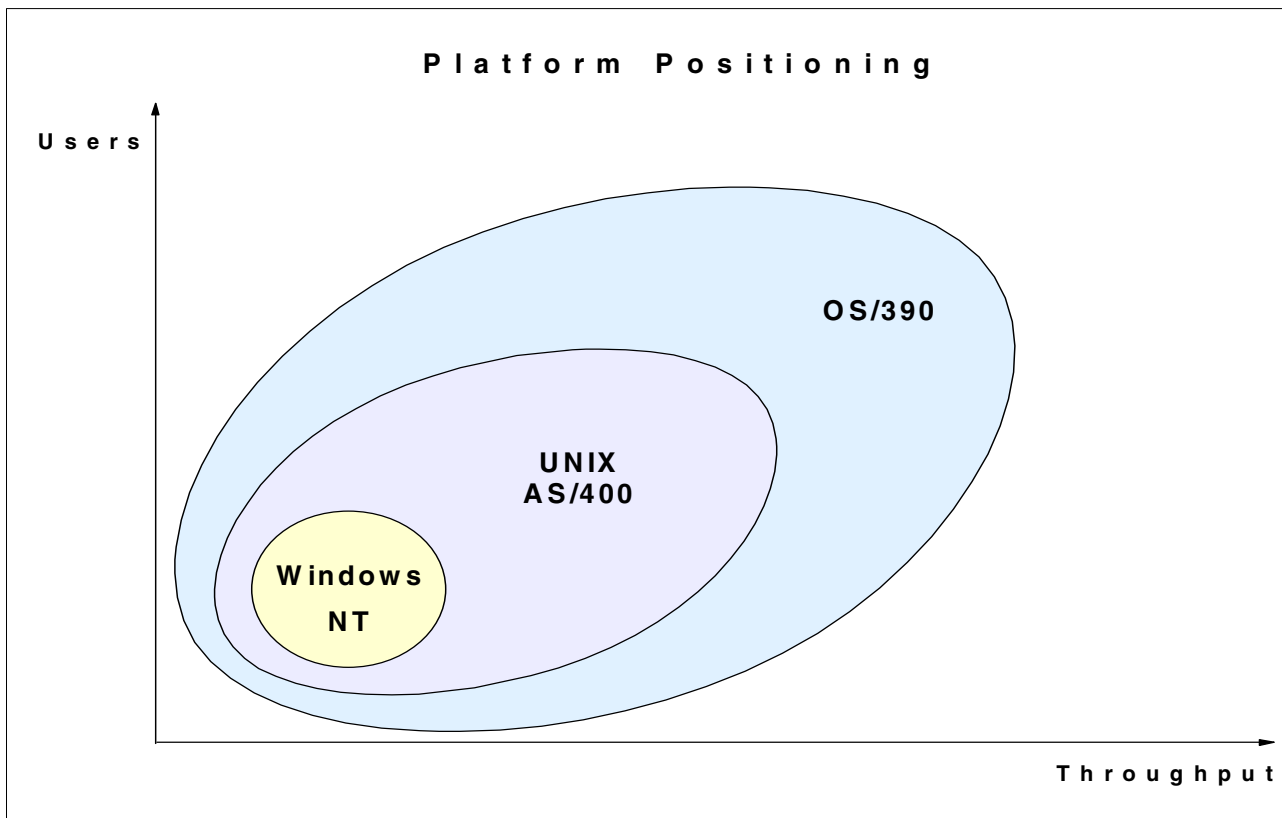


Figure 3. Qualitative sizing of Net.Commerce solutions across different platforms

2.3 Differences between solution architectures

When architecting the Net.Commerce system, there are some important differences to be aware of.

The first is that there will be fewer servers in the OS/390 solution. For example, a typical high-level drawing of a Windows NT or UNIX solution will look something like the one shown in Figure 4 on page 13.

Common Windows NT or UNIX Configuration

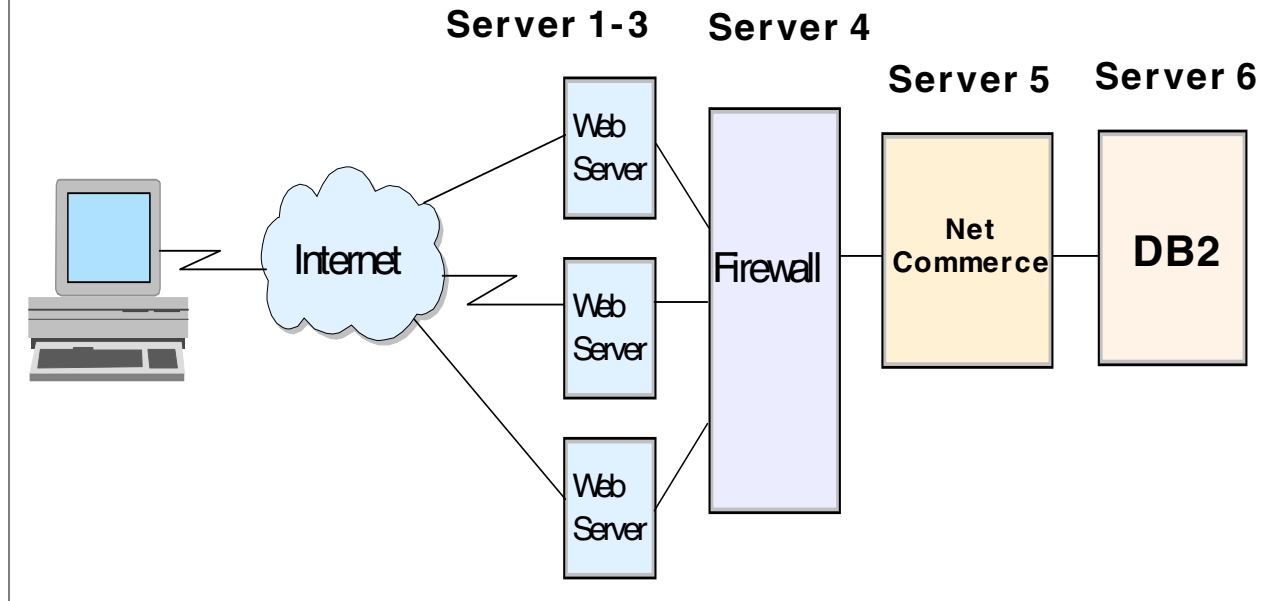


Figure 4. Typical Windows NT or UNIX Net.Commerce system

Why is the Web server on the opposite side of the firewall as the Net.Commerce server? This is because some UNIX and Windows NT Web servers are known to be open to attack. A hacker, using buffer overrun techniques, for example, could attack and gain control of some Web servers. If the Web server were inside the firewall, a hacker could then reach (attack) other machines inside that network.

If the hacker does manage to take control of the Web server, the hacker will have achieved, at a minimum, a “Denial of Service” attack (for example, the Web server is down, so nobody can use the site).

Why are the Net.Commerce and database servers on separate machines? This is for performance and scalability reasons.

In addition the customer may be using the Net.Commerce staging server function, which typically means that they will have more than one database server.

Common OS/390 Configuration

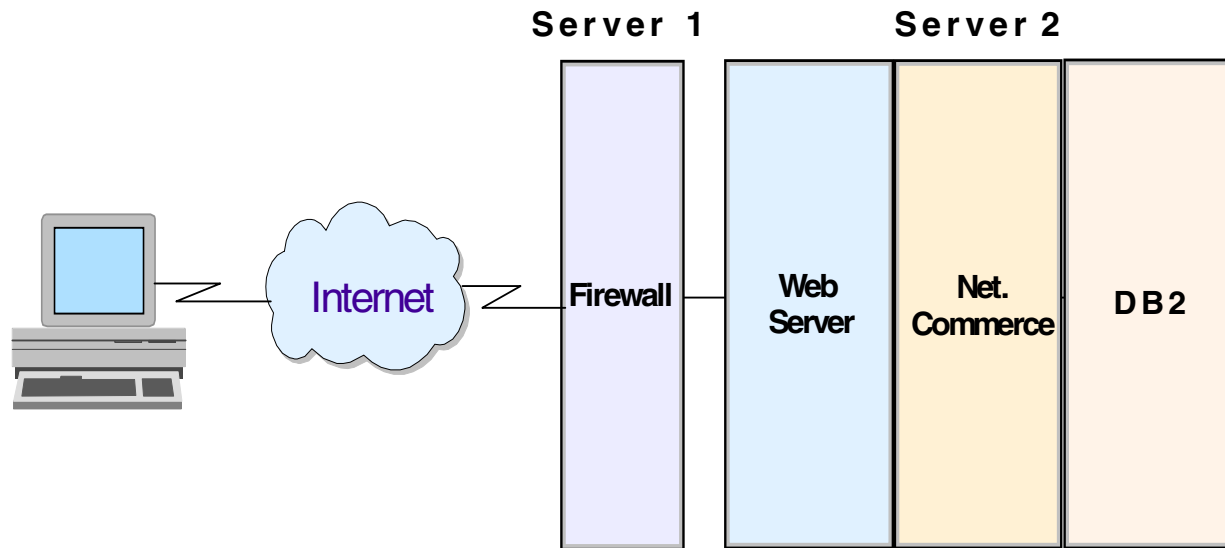


Figure 5. Typical OS/390 Net.Commerce system

In the OS/390 chart shown in Figure 5, you notice the following differences: the Web server resides on the same machine as the Net.Commerce server, inside the corporate network. In addition, the Net.Commerce and database servers reside on the same machine.

IBM has tested the OS/390 Web server with buffer overflow attack and the typical results of the hacker ending up in root mode or in control of the server was not realized. Server misguidance, where the hacker enters some parameters in an attempt to cause the Web server to do things that are not desirable, is very difficult to do and, if the Web server (and its infrastructure) is configured correctly, should not be possible at all.

Currently, with OS/390 Net.Commerce, the commerce server must reside on the same OS/390 system as the Web server. This is because the OS/390 Web server must communicate with Net.Commerce over a local socket.

The database server must reside on the same system as the commerce server. However, this database server can be linked to remote DB2 servers using existing techniques, such as distributed relational database architecture (DRDA).

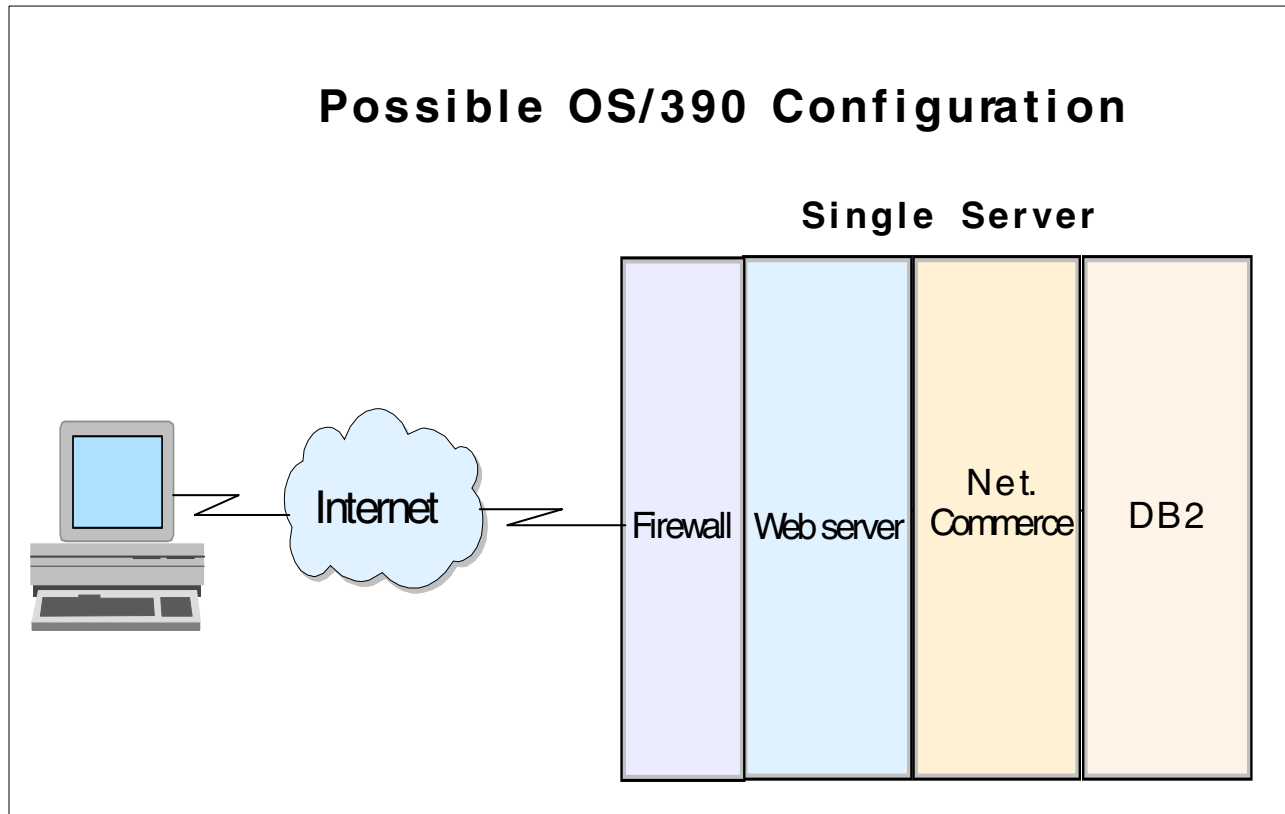


Figure 6. Possible OS/390 Net.Commerce configuration

The chart shown in Figure 6 is very similar to the previous chart except that this chart shows that it is also possible to run the firewall on the same server as the Web server, the Net.Commerce server, and the database server.

Alternatively, a customer might choose to have UNIX or Windows NT servers managing the HTTP traffic between the shopper and the site and to forward Net.Commerce URLs through to Net.Commerce running on OS/390.

2.3.1 Differences in staging environments

Once the site is in production, you will want to test changes such as new prices, updated product descriptions, new graphics, and templates in a test environment before putting the changes into production. This gives you a chance to correct any errors that occur when updates are made and allows you to visually judge and review your modifications before you present them to customers.

2.3.2 Windows NT and UNIX testing environment

Net.Commerce on the Windows NT and UNIX platforms has a function known as *staging server*. The staging server database is a separate database that you make your changes to. Once you are happy with the changes, they can be migrated to the production database.

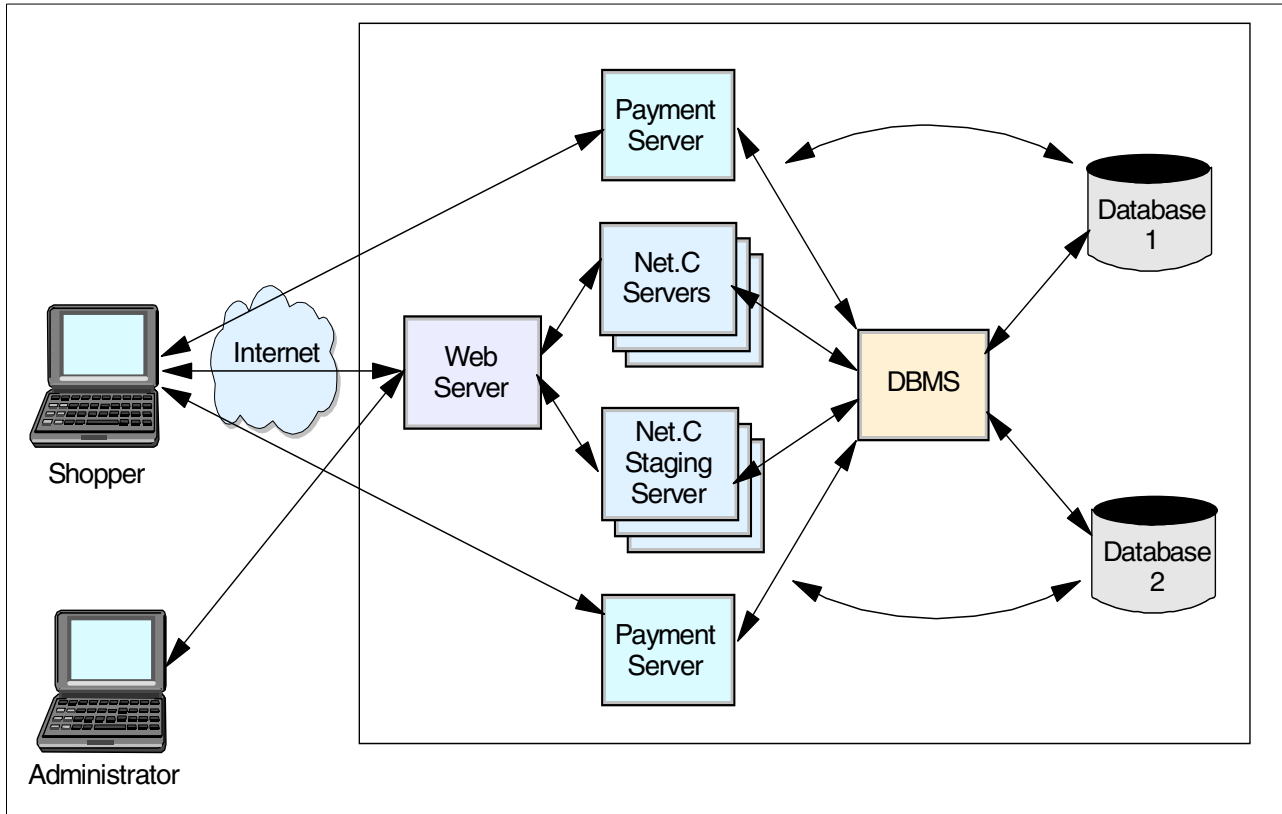


Figure 7. Windows NT and UNIX staging environment

2.3.3 OS/390 testing environment

Net.Commerce for OS/390 does not have the staging server function. The testing function with OS/390 Net.Commerce is provided by running a test environment in parallel to the production environment. To avoid any adverse effects to your production environment, it is recommended that you run this parallel environment in a separate logically partitioned mode (LPAR) or as a guest machine under VM/ESA.

There are a number of ways to move the test data into production. For instance, you could copy the HFS files (HTML pages, Net.Data macros, images, dlls, and so on) from the test environment to the production environment. A second approach would be to switch the Domain Name Server (DNS) to change the test to the production system and vice versa. The second approach is more complex once the backend system connections are taken into account, so customers tend to adopt the first approach (copying) to install changes to their production environment.

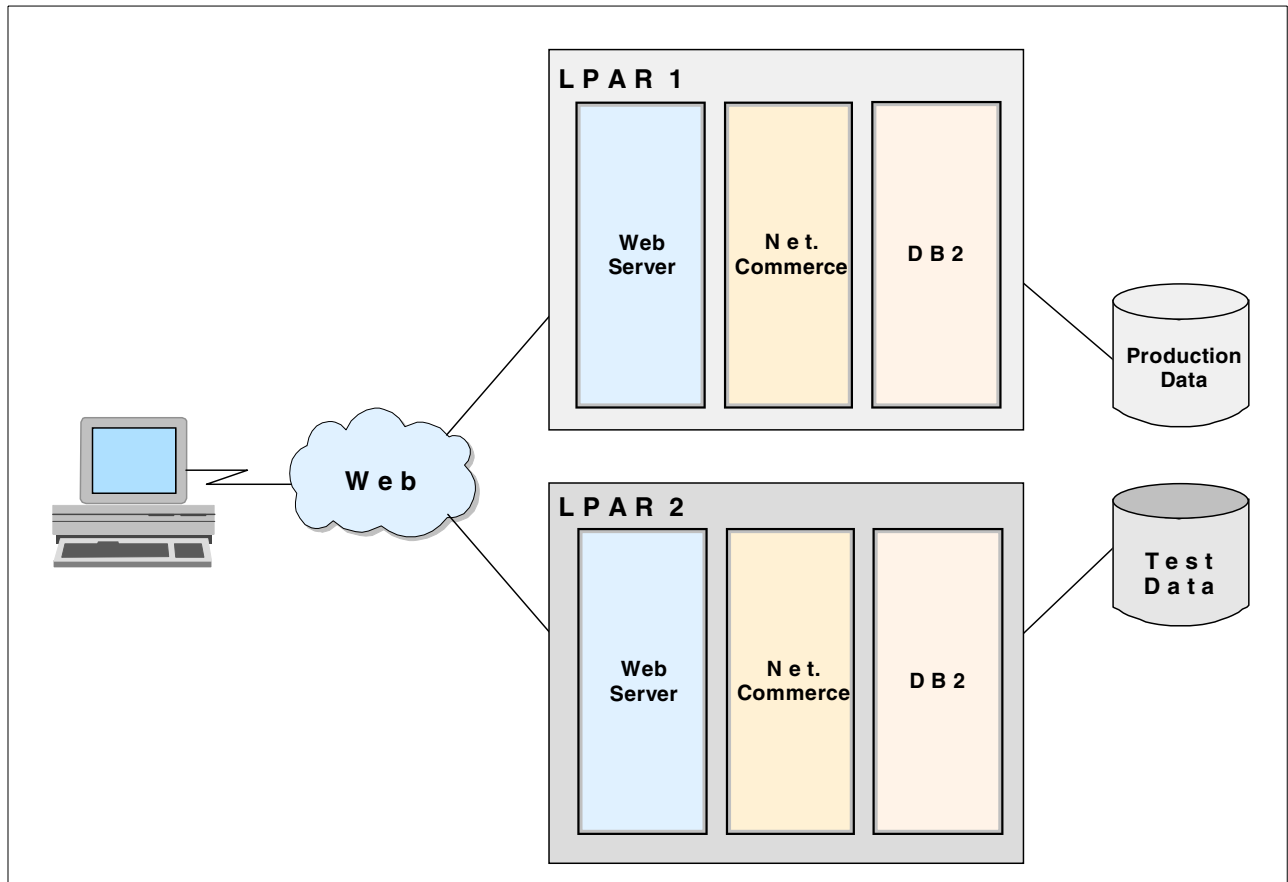


Figure 8. OS/390 testing environment

Chapter 3. General migration considerations

This chapter covers the general migration issues when moving from Net.Commerce on Windows NT or UNIX to Net.Commerce OS/390. Here, we cover migrating HTML files and images, Net.Data macros, and C++ commands and functions. The most effort you will spend when moving your site is the migration of the Net.Commerce database, which is covered in Chapter 4, “Migrating the database” on page 49.

3.1 ASCII to EBCDIC

IBM defined extended binary coded decimal interchange code (EBCDIC) as its character encoding scheme (8 bits for a character). The American National Standards Institute (ANSI) defined a different code called American National Standard Code for Information Interchange (ASCII) that uses 7 bits for a character. All UNIX and PC systems use ASCII in one form or another, but IBM OS/390 and AS/400 servers, among others, continue to use EBCDIC.

Porting from an ASCII to an EBCDIC platform can be a significant concern when moving applications to OS/390. For the latest information available, please review the following sources of information:

- <http://www.s390.ibm.com/oe/bpxa1por.html>
- *Porting Applications to the OpenEdition OS/390 Platform*, GG24-4473, published in April 1995.

When debugging, consider whether the problem could be an ASCII to EBCDIC translation issue.

3.1.1 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 *lex* 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 leftmost 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 is greater than a, but in EBCDIC, a is greater than A.

For further information, please refer to the document at:

<http://www.s390.ibm.com/products/oe/bpxa1p03.html>

3.1.1.1 ASCII-like application environment

A library called LIBASCII has been developed for use by programmers who are accustomed to ASCII programming (this library file is available from <http://www.s390.ibm.com/oe/bpxa1toy.html>). Therefore, if a program is compiled to

ASCII using this layer, all parameters and return values for character-based system calls, for example, `setenv()`, `printf()`, and so forth, will be translated to and from EBCDIC as required.

Note that 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 less than A, 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 system, 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 of the usage of the LIBASCII tool, please refer to Section 6 of *Porting a UNIX Application to OS/390 UNIX: Problems Encountered and Lessons Learned* available at: <http://www.s390.ibm.com/oe/aixdb2/aixdb2.html>

3.1.1.2 Commands and functions

Certain OMVS shell commands (`iconv` and `pax`), TSO/E commands (`OPUT`, `OGET`, and `OCOPY`), and C functions (`__atoe()`, `__atoe_l()`, `__etoa()`, and `__etoa_l()`) handle ASCII to EBCDIC conversion.

Refer to <http://www.s390.ibm.com/products/oe/bpxalp03.html> for further information about ASCII to EBCDIC porting issues.

3.1.2 File transfer using TCP/IP FTP

We assume TCP/IP is installed on both the workstation and the OS/390 system, so you can use the file transfer protocol (FTP) to transfer your source data to OS/390. If transferring single-byte data (ASCII mode), FTP will convert the data from ASCII to EBCDIC for you. Binary data can also be sent using FTP if binary mode is specified.

We recommend the use of a full screen FTP file transfer tool, such as `ws_ftp`, which can be downloaded from: <http://www.download.com>

When using `tar` or `pax` to move files from UNIX to OS/390, move the ASCII files into one archive and the binary files into another. Then expand the archives on OS/390 using the following commands:

```
pax -rf ascii_archive.pax -o from=iso-8859-1,to=ibm-1047
pax -rf binary_archive.pax
```

3.1.3 Net.Commerce issues

HTML files should be relatively simple to migrate from Windows NT or UNIX to OS/390. One consideration to remember is that 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”.

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.

3.1.4 Shopper password porting

The password information for shoppers is stored in the Net.Commerce database in the *shopper* table. Since the information is stored in encrypted format, a binary port of the password will not give us the correct representation on OS/390. An ASCII to EBCDIC conversion of the password while porting would work provided the encryption algorithms used across the platforms are the same. But, unfortunately, at the present time, the encryption algorithm used on OS/390 is not the same as that used on Windows NT or UNIX.

At the time of writing this book, an update to the Net.Commerce code is being written that will allow Net.Commerce sites running on Windows NT or UNIX to port the shopper's password to OS/390. For full details of the code change and how to implement it, please see OS/390 Net.Commerce APAR PQ32922.

3.2 Net.Data compatibility

Net.Data capability and macro format is very similar between the different platforms. However, there are some differences that require attention.

Unlike HTML files, which may be stored in the HFS directory in ASCII, Net.Data macros must be stored in EBCDIC. One problem is the use of tabs for formatting macros on a PC—these tabs are not supported by the ISPF editor on OS/390. It is very important to test all of your macros, perhaps through the use of the Net.Commerce ExecMacro command, after you port them to OS/390. For example, the command to test a macro called ported.d2w held in the Net.Commerce root directory is:

```
http://<os/390_server>/cgi-bin/ncommerce3/ExecMacro/ported.d2w/report
```

The schematic shown in Figure 9 on page 22 is a visual representation of the location of various Net.Data macros in an existing Windows NT Net.Commerce implementation and where they might move to in an OS/390 Net.Commerce environment.

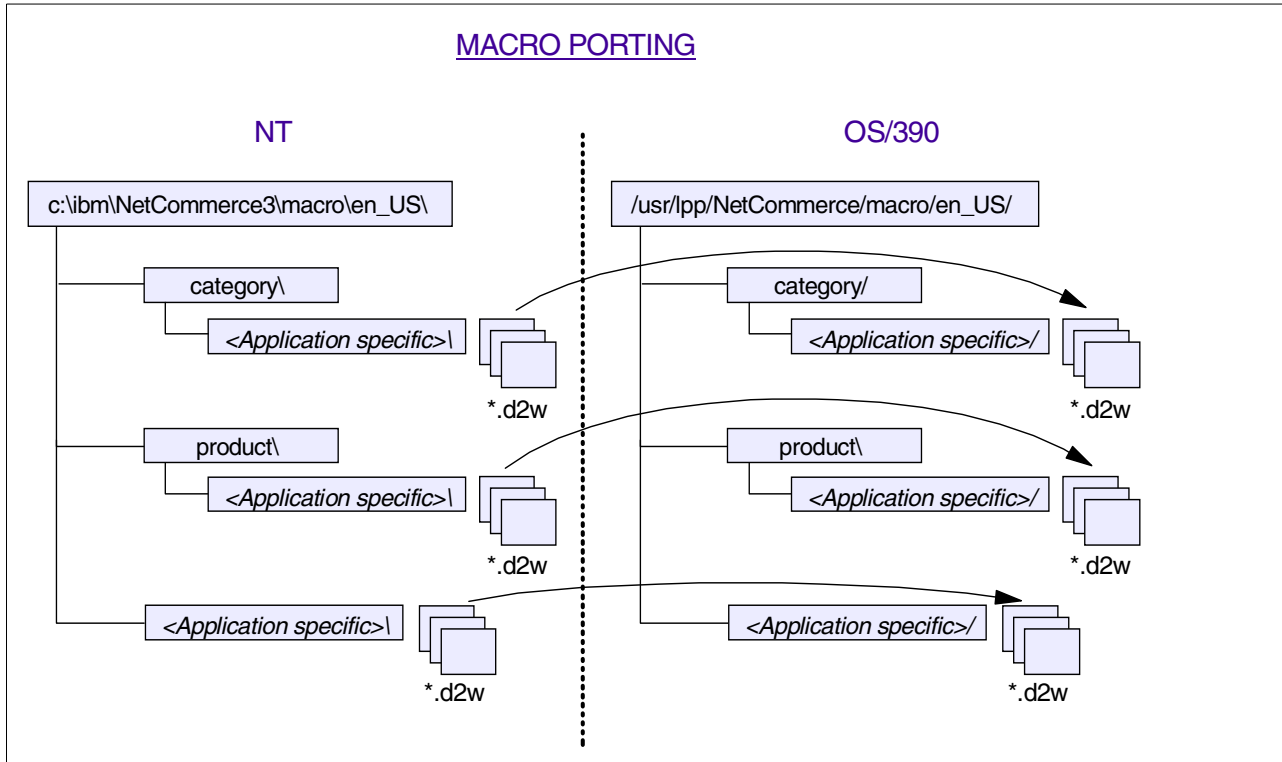


Figure 9. Porting Net.Data macros from Windows NT to OS/390

3.2.1 Path names in Net.Data macros

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 drive letters removed.

3.2.2 DTW_ODBC must be changed to DTW_SQL

The Windows NT and UNIX versions of Net.Commerce generally use ODBC. However, on OS/390, you must change all occurrences of DTW_ODBC to DTW_SQL in your Net.Data macros. DTW_ODBC is not as performant as DTW_SQL on OS/390 and to get DTW_ODBC to work at all on OS/390 is non-trivial.

3.2.3 Using the UNIX sed command for global changes

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
```

The `<` and `>` are literal, so you need to type them in.

To change DTW_ODBC to DTW_SQL in the same file, the command is:

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

To change all files in a directory you could use a UNIX shell loop, for example:

```

for a in * <press the enter key>
do <press the enter key>
cp $a $a.0 <press the enter key>
sed 's/DTW_ODBC/DTW_SQL/' < $a.0 > $a <press the enter key>
done <press the enter key>

```

3.2.4 Net.Data special characters

Your existing Net.Data macros may contain hidden characters, such as tabs, that do not create a problem on Windows NT, but do on OS/390. 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.

OS/390 Net.Data macros should not contain the `\t` or tab character. The tab has to be changed to a space or null. For example, the `\t` character in the following SQL has to be changed to a space:

```

%function(dtw_sql) GET_1800NUMBER() {
  select mecph2
  \t from   merchant
  \t where  merfnbr=$(MerchantRefNum)
  %REPORT {
    ....
  }
  %MESSAGE{100:{ %} :continue %}
}

```

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, 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)">

```

This should be changed to:

```

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

```

3.2.5 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. See Table 2.

Table 2. 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

Category	What's not supported for OS/390
Web Registry Functions	There are no Net.Data Web registry functions that apply to OS/390

If your Net.Data macros on Windows NT or UNIX utilize any of these functions, you may need to modify the macros to work on OS/390. Please refer to the Net.Data Reference manual for more details:

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

This manual is available on the Web in HTML and PDF formats at the above URL.

3.3 HTML and image considerations

HTML pages should be transferred to the OS/390 server using a file transfer program such as ftp in ASCII mode, and the image files (gif, jpeg, and so on) should be transferred to the OS/390 server using a file transfer program, such as ftp in binary mode.

The following graphic details the areas where you will find the HTML and gif files on Windows NT and where they should go on the OS/390 server.

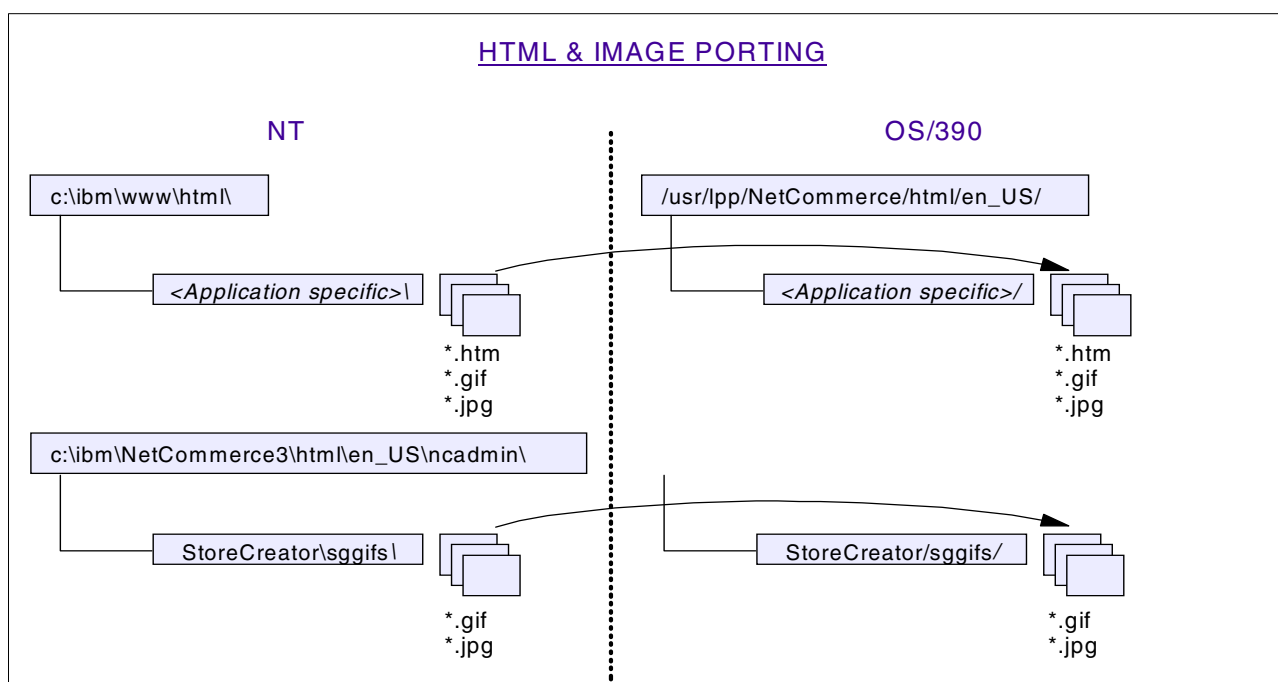


Figure 10. HTML and image porting

OS/390 UNIX Systems Services is case sensitive, so care will have to taken to ensure that files are stored in their correct case in the OS/390 file system. For example, Graphics Interchange Format (GIF) may be stored as *.GIF on Windows NT, and if they are referred to as *.gif in the DB2 tables and HTML files of an Windows NT Net.Commerce system, they will be found. This is not the case for OS/390. OS/390 UNIX Systems Services is case sensitive and these files would have to be renamed as *.gif to allow them to be resolved correctly.

These files are referenced by the `Pass` directives in the Web server's `httpd.conf` file. A `Pass` directive looks like the following:

```
Pass /demomall/* /usr/lpp/NetCommerce/html/en_US/demomall/*
```

This means that whenever the Web server receives a URL request with the string `demomall` immediately following the host name, it resolves the location to the full path as specified by the `Pass` directive.

Therefore, if you have your HTML files in a directory called `/usr/lpp/NetCommerce/html/en_US/myhtml`, and image files in a directory called `/usr/lpp/NetCommerce/html/en_US/myimages`, and you want to refer to them just as `myhtml` and `myimages`, you would need to include two `Pass` directives in the `httpd.conf` file:

```
1. Pass /myhtml/* /usr/lpp/NetCommerce/html/en_US/myhtml/*
2. Pass /myimages/* /usr/lpp/NetCommerce/html/en_US/myimages/*
```

Do not put these `Pass` directives in the directives added by the `Net.Commerce` configuration utility `CMNCONF`. The statements added by `CMNCONF` are located at the top of the Web Server configuration file (`httpd.conf` by default). `CMNCONF` overwrites this section in the Web server configuration file every time you modify an existing instance or create a new instance with `CMNCONF`. These directives are between these two comment blocks in the Web server configuration file:

```
##### IBM Net.Commerce #####
... various statements ...
##### End of IBM Net.Commerce #####
```

To activate the changes, you must restart the Web server each time its configuration file is modified. You may issue the following OS/390 console command to restart the Web server:

```
F <web server task name>, APPL=-RESTART
```

For more information about running the `CMNCONF` utility, please refer to 3.11, “Configuration” on page 35.

3.4 Product Advisor compatibility

Product Advisor runs on all of the `Net.Commerce` platforms. There are some extra steps that have to be taken when using Product Advisor with an OS/390 target database and these are documented in the manual *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862. Porting data held in existing `Net.Commerce` Product Advisor database tables on Windows NT or UNIX to Product Advisor tables on OS/390 was not carried out as part of this redbook.

3.5 Catalog Architect compatibility

The output from the Catalog Architect tool is not compatible with the current release of `Net.Commerce` on OS/390, and, therefore, should not be used without modifications.

3.6 Payment Server

Payment Server runs on all of the Net.Commerce platforms. Porting data held in existing Net.Commerce Payment Server database tables on Windows NT or UNIX to Payment Server tables on OS/390 was not carried out as part of this redbook.

The Payment Server on the distributed platforms has been replaced by the Payment Manager.

3.7 Backend integration

There are tools available for integrating an OS/390 Net.Commerce V3.1.1 site to various existing backend fulfillment systems (these have not been tested at the 3.1.2 level). IBM has made enterprise integration samples available for connecting Net.Commerce to CICS, IMS, MQSeries, and electronic data interchange (EDI) servers along with miscellaneous utilities. The sample code is available on the Web at:

<http://www.s390.ibm.com/nc/ecommerce/sampcode.html>

The IBM-supplied function, available on the Web, is structured as follows:

- CICS and IMS integration – Two high-level communication functions for executing transaction programs on both CICS and IMS. Each function will allow you to specify all input data for a given transaction program and receive all of the output in a single call.
- EDI integration – A set of functions that permit the collection of purchase order information from the Net.Commerce system and the exchange of this information and responses with an EDI service provider.
- MQSeries integration – A set of functions that allow the exchange of information with an MQSeries application program.
- Asynchronous server – A separate server daemon to manage information exchange that is asynchronous to the Net.Commerce Server daemon operation.
- Miscellaneous utilities and samples including:
 - A family of functions that perform data transformation, for example, integer to zoned decimal and packed decimal to integer for communicating with transaction programs that are written in COBOL.
 - A set of 19 functions that are skeleton replacements for the existing Net.Commerce server API functions. Some functions are populated with sample calls to the CICS, IMS, and MQSeries communication functions.
 - Exception macros for processing errors encountered in the API task functions.
 - Documentation in HTML for online viewing and PostScript for hardcopy that describes how to make use of the supplied functions.

3.7.1 CICS integration

We have customers today integrating Net.Commerce on OS/390 with CICS fulfillment systems running on OS/390.

The chart in Figure 11 shows a real customer implementation in the Nordic region of Europe. This is a good example of putting a new Web-based channel on an existing telephone-based order system.

Inventory, order, and customer information is kept in the customer fulfillment CICS system. Net.Commerce connects to a skeleton CICS region running in the Web LPAR through the Common Gateway Interface (CGI). The CICS running in the Web LPAR connects to CICS running in the production LPAR through CICS to CICS multi-region option (MRO) communication.

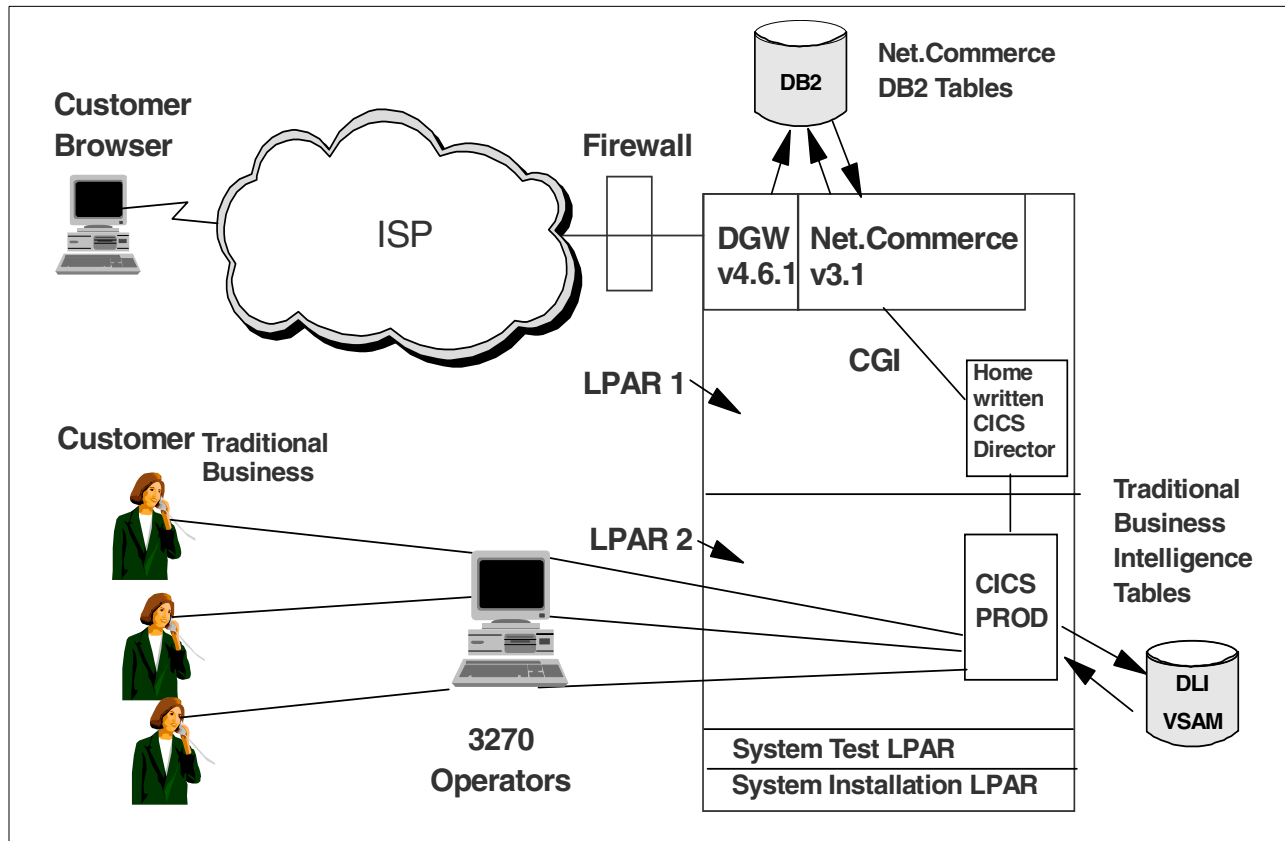


Figure 11. Connecting to backend CICS

3.7.2 IMS integration

IMS integration can be achieved through IMS connectors. Further details can be found at: <http://www.ibm.com/software/data/ims/imswwwc.html>

For more information about the Windows NT and UNIX integration with IMS refer to: <http://enterprise.torolab.ibm.com/~rintjema/netcom3/refs/srimfile.htm> This document outlines the files that you may have used when you originally integrated Net.Commerce (Windows NT or UNIX) with IMS.

3.7.3 Commerce Integrator

While Net.Commerce for OS/390 provides sample code as mentioned, the IBM Commerce Integrator product is not available for OS/390. This is not to be confused with the MQSeries Integrator that is available for OS/390.

The IBM Commerce Integrator available with the distributed Net.Commerce platforms enables integration with Enterprise Resource Planning (ERP) and other backend systems and supports sending and receiving messages using standard MQSeries messaging. The adapter focuses on data connectivity across systems and allows systems to communicate and share data (which is contained within the electronic messages) to complete business processes. The Net.Commerce MQSeries adapter works with a set of outbound and inbound messages that help integrate Net.Commerce business processing with ERP and other backend-system business processing. In addition, the IBM Commerce Integrator supports message extension and new messages.

For further information about the IBM Commerce Integrator, please see:

IBM Commerce Integrator User's Guide for AIX, SC09-2878

IBM Commerce Integrator User's Guide for Windows NT, SC09-2865

IBM Commerce Integrator User's Guide for Sun Solaris, SC09-2879

For additional information about backend integration, refer to the IBM Redbook *Integrating Net.Commerce with Legacy Applications, SG24-4933*.

3.8 Common Connector Framework

IBM also provides the Common Connector Framework (CCF) with a complete family of connectors for existing IMS, CICS, and DB2 MQSeries applications. The connectors support multiple languages like Java, HTML, C/C++, COBOL, and REXX. They offer a broad range of alternatives that allow companies to determine what is best for their environment based on the skills, performance, and security requirements they have.

These products support the IBM WebSphere strategy using Java and the Common Connector Framework; notably, CICS with VisualAge for Java have been early supporters of the CCF.

3.9 Commands, Tasks, and Overridable 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 12 on page 29 shows a summary chart of Commands, Tasks, and Overrideable Functions and how they relate to each other.

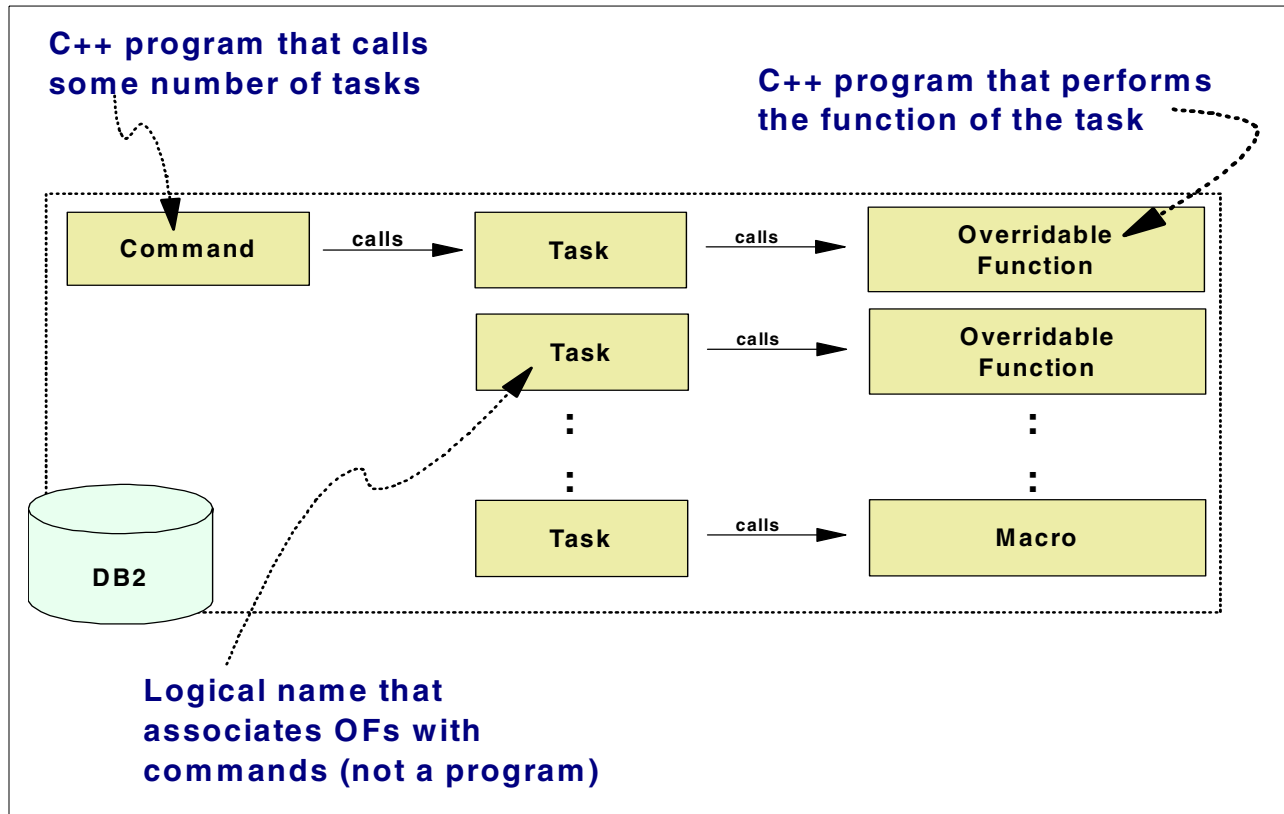


Figure 12. Overview of Commands, Tasks, and Overridable Functions

3.9.1 Commands and Overrideable Functions

Commands and Overrideable Functions are currently implemented in C++ shared objects (known as dynamic link libraries on Windows NT).

The C++ source from your Windows NT or UNIX Net.Commerce application will have to be recompiled on OS/390. See Chapter 5, "Application development" on page 77 for details of how to recompile your C++ Net.Commerce programs on OS/390.

The location and names of Net.Commerce executables varies between Windows NT, UNIX, and OS/390. On OS/390, the Command or Overrideable Function will have two components, a *.so file (the shared object code) and a *.x file (the side deck). On Windows NT, there is only one component, a *.dll file.

You place the *.so file and the *.x file in the /usr/lpp/NetCommerce/lib directory by default. It is advisable, though, that you place them in a user code directory such as /u/code. You then add this directory to the LIBPATH statement in the <Net.Commerce instance name.envvars> file, for example:

```
PATH=/bin:../usr/lpp/internet/bin
SHELL=/bin/sh
TZ=EST5EDT
LANG=en_US
NLSPATH=/usr/lpp/NetCommerce/msg/%L/%N:/usr/lpp/NetCommerce/msg/en_US/%N
LIBPATH=/usr/lpp/internet/bin:/u/code:/usr/lpp/NetCommerce/lib
STEPLIB=CMN.V312.SCMNMOD:MYCODE.LOADLIB
```

You can optionally copy your C++ Commands and Overrideable Functions from the HFS to OS/390 data sets and have Net.Commerce access them from OS/390 data sets rather than the HFS. The sample makefile in Appendix D.1, “OS/390 Net.Commerce sample makefile” on page 97 shows you how to do this. If you go the OS/390 route, you should remove the *.so file and put a link in the HFS to link externally to the code in the OS/390 data set. This can be achieved with the following OS/390 UNIX shell commands:

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

The name MYCODE must be the name of the OS/390 data set member that you copied the *.so as. The OS/390 data set containing the MYCODE member must be in one of:

- <net.commerce instance.envvars> on the LIBPATH statement
- Net.Commerce startup procedures STEPLIB
- OS/390 system linklist or LPA

Just placing the data set name on the LIBPATH statement is recommended. 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.

There are potentially slight performance improvements in placing your C++ code in OS/390 data sets and then placing these data sets in the LINKLIST or LPA.

Most Net.Commerce for OS/390 customers run their Commands and Overrideable Functions directly from the HFS and not from OS/390 data sets.

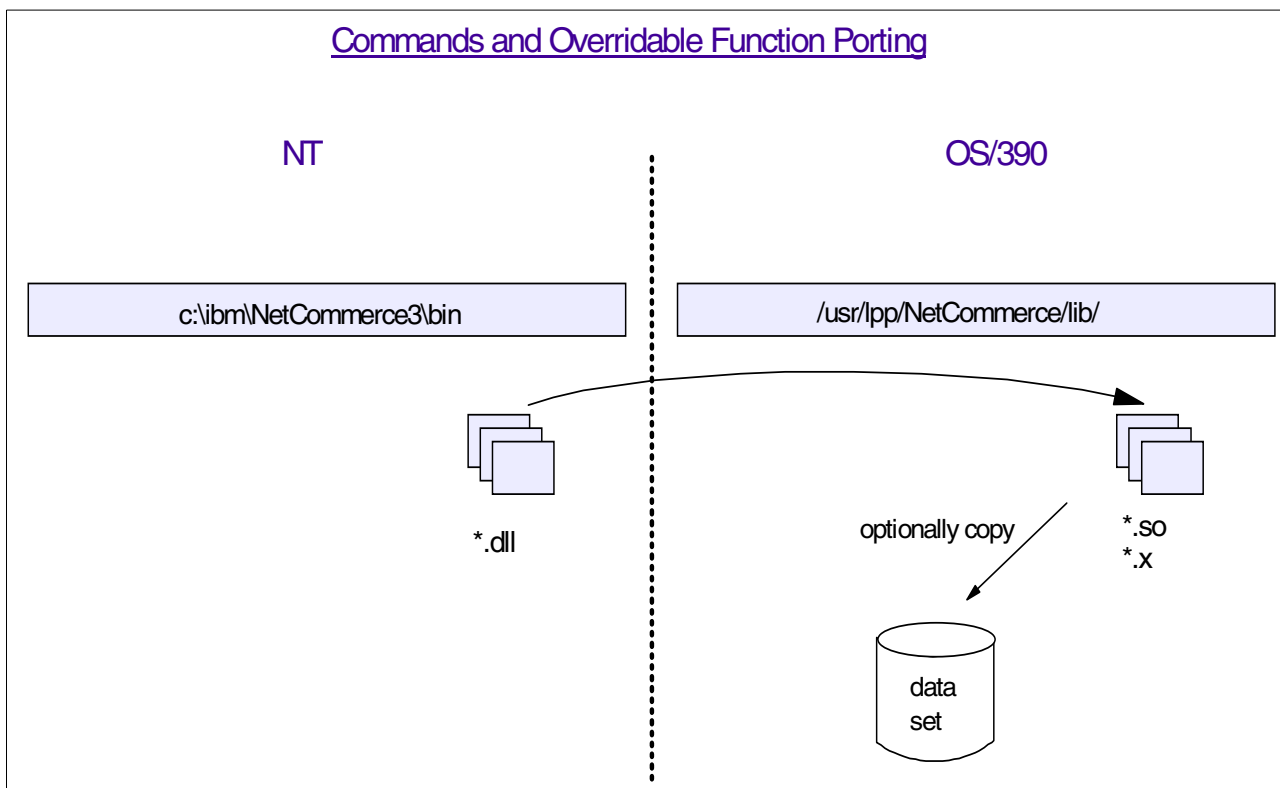


Figure 13. Migration of Net.Commerce Commands and Overrideable Functions

When porting the Net.Commerce database from an Windows NT or UNIX site to OS/390, you will insert information into the Net.Commerce CMDS table (among others) on OS/390. See Chapter 4, “Migrating the database” on page 49 for details of how we recommend you migrate your existing Net.Commerce database to OS/390. If you are porting from Windows NT, by default our process means that you will insert the name of any Commands you may have written into the DLL_NAME column of the CMDS table:

```
<command_name>.dll
```

On OS/390, the DLL_NAME column should contain:

```
<command_name>.so
```

Therefore, you should edit the CMDS table to change the name from *.dll to *.so. This can be achieved with SQL, such as:

```
update <owner>.CMDS set DLL_NAME='<command_name>.so' where REFNUM=xxx
```

You can find the reference number (REFNUM) for the Net.Commerce Command from an SQL query, such as:

```
select * from <owner>.CMDS where DLL_NAME='<command_name>.so'
```

3.9.1.1 MAX(REFNUM) in the CMDS table

In the Windows NT or UNIX Net.Commerce default databases (including demomall), the maximum value for REFNUM in the CMDS table is lower than the maximum value for REFNUM in the CMDS table on OS/390. This is because the OS/390 implementation has extra Commands when compared to Windows NT or UNIX implementations (for example, the ExecDB2Macro Command). This means that any custom-written Commands that you have registered on Windows NT or UNIX will need to be re-registered with higher REFNUM values on OS/390.

3.9.1.2 MAX(REFNUM) in the OFS table

From looking at the OFS table, we see that OS/390 Net.Commerce has more Overrideable Functions than the Windows NT or UNIX implementations, so you will have to re-register your OFS with higher REFNUM values than those used on Windows NT or UNIX.

Edit the OFS table to change the REFNUM. This can be achieved with SQL, such as:

```
update <owner>.OFS set REFNUM=xxx where DLL_NAME='<command_name>.so'
```

You can find the reference number (REFNUM) for the Net.Commerce Command from an SQL query, such as:

```
select * from <owner>.OFS where DLL_NAME='<command_name>.so'
```

3.9.2 Tasks

There are three kinds of Tasks:

- Process Task – This is a Task that runs some program to do some work, such as fetch the base price of a product, or determine the inventory level.
- View Task – This is a Task that runs a Net.Data macro.

- **Error Task** – These are actually a special kind of View Task that are associated with Process Tasks. If the process encounters an error, this Task is called so that the shopper can be presented with a screen informing them of the error.

User-written Tasks will be ported to OS/390 when you port your Net.Commerce database to OS/390 as described in Chapter 4, “Migrating the database” on page 49.

There are many tables associated with Tasks, such as MACROS, TASKS, TASK_MER_OF, and OFS. You should review each of these tables to ensure that the integrity of your Commands, Tasks, and Overridable Functions is maintained during the port to OS/390.

Note that OS/390 does not have an OF_TASK table which is found in versions of Net.Commerce on the distributed platforms.

3.9.2.1 MAX(TKRFNBR) in the TASKS table

Similarly, MAX(TKRFNBR) is lower in the Windows NT and UNIX versions of the Net.Commerce TASKS table, so you will have to re-register your custom-written Tasks in the OS/390 Net.Commerce TASKS table with higher TKRFNBR numbers than used in Windows NT or UNIX.

3.9.2.2 MAX(TKRFNBR) in the TASK_MER_OF table

Similarly, MAX(TKRFNBR) is lower in the Windows NT and UNIX versions of the Net.Commerce TASK_MER_OF tables, so you will have to reassign your custom-written Overrideable Functions to their Tasks in the TASK_MER_OF table with higher TKRFNBR numbers than used in Windows NT or UNIX.

3.9.2.3 MAX(TKRFNBR) in the MACROS table

MAX(TKRFNBR) is also lower in the Windows NT and UNIX versions of the Net.Commerce MACROS table, so you will have to reassign your custom written Tasks to their Net.Data macros in the MACROS table with higher TKRFNBR numbers than used in Windows NT or UNIX.

3.10 Installation and operation differences between platforms

On all platforms, the installation process consists of the following steps:

1. Check for the minimum hardware and software requirements.
2. Ensure that the operating system is installed correctly with any required fixes and the correct drivers.
3. Ensure that TCP/IP is configured correctly including name resolution.
4. Ensure that no other Web server is installed and listening on the HTTP and SSL ports (80 and 443 respectively). For example, on Windows NT, make sure that Microsoft IIS is not installed or is disabled.
5. Create a user for the database owner and administrator.
6. Install the software from the Net.Commerce CD-ROM or product tapes. This activity is highly platform specific. On Windows NT, the Install Shield Wizard is used for installation. On UNIX, db2setup is used to install DB2 Universal Database and the System Management Interface Tool (SMIT) is used to install the remaining components. On OS/390, system modification program/extended (SMP/E) is used to install products from tape.

7. Configure a Net.Commerce instance using the platform-specific configuration managers (for example, a Java applet for UNIX and Windows NT and a TSO/ISPF application for OS/390).
8. Create a self-signed SSL certificate for testing or obtain a production SSL certificate from a Certificate Authority, such as VeriSign or Thawte.
9. Enable the Java Database Connectivity (JDBC) server if you will be using the Net.Commerce Product Advisor.

The Net.Commerce software for Windows NT and UNIX is shipped on CD-ROM. The OS/390 version of Net.Commerce is shipped on a product tape. There are two FMIDs delivered on the tape:

- H01B312 (Net.Commerce V3.1.2)
- HCME121 (Payment Server)

The Payment Server needs to be installed only if you are planning on using the Secure Electronic Transactions (SET) payment option available with Net.Commerce.

A high-level overview of the installation and configuration process for UNIX, Windows NT, and OS/390 are shown in Figure 14, Figure 15 on page 34, and Figure 16 on page 34.

3.10.1 AIX installation overview

Figure 14 shows an overview of the AIX installation.

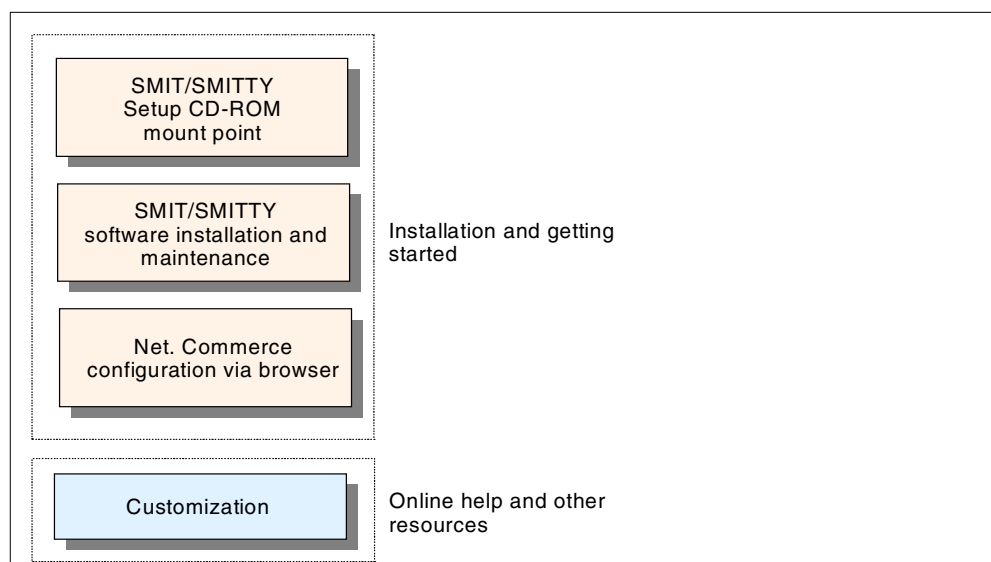


Figure 14. A high-level overview of installation and configuration on AIX

Refer to *IBM Net.Commerce for AIX Installing and Getting Started*, GC09-2627 for further details.

3.10.2 Windows NT installation overview

Figure 15 on page 34 shows an overview of the Windows NT installation.

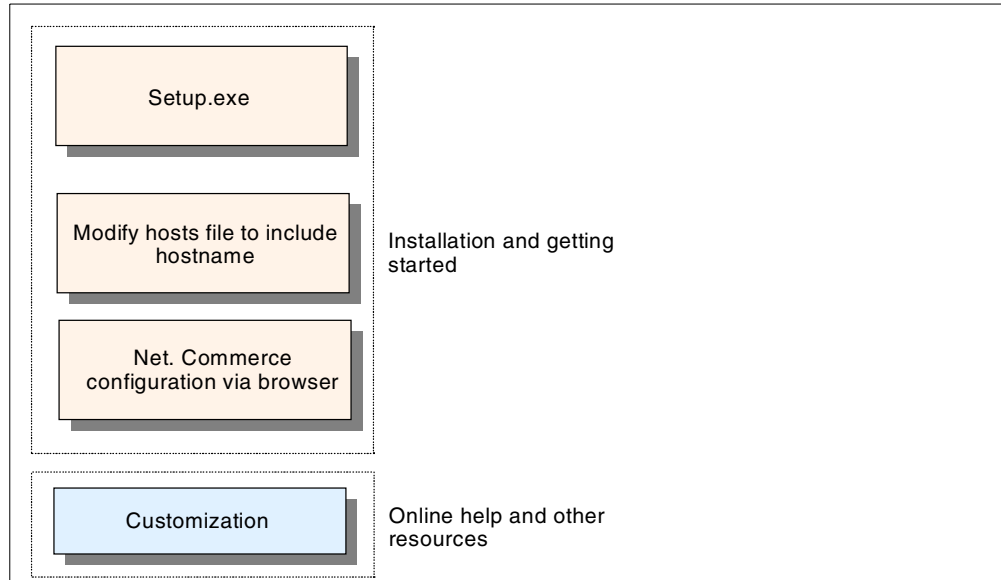


Figure 15. A high-level overview of installation and configuration on Windows NT

Please refer to *IBM Net.Commerce for Windows NT Installation and Getting Started*, GC09-2626 for further details.

3.10.3 OS/390 installation overview

Figure 16 shows an overview of the OS/390 installation.

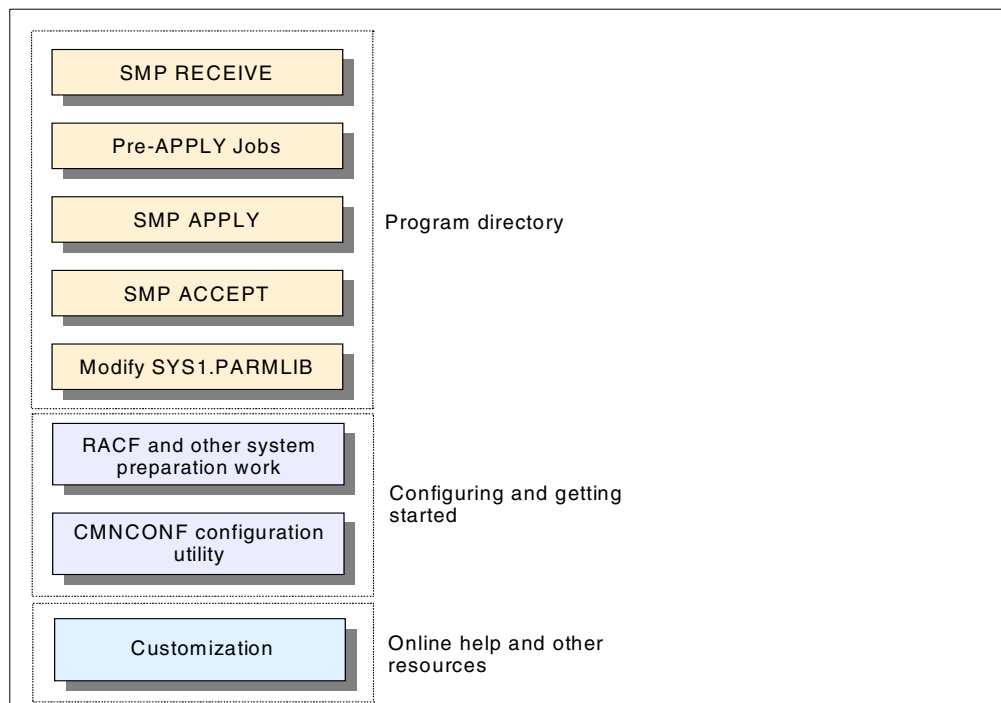


Figure 16. A high-level overview of installation and configuration on OS/390

Please refer to *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862 for further details.

3.11 Configuration

Once Net.Commerce has been installed (on any platform), the Web server, DB2, and Net.Commerce itself need to be configured before the site can be used. During the configuration process, the Web server configuration file is updated, the DB2 schema is created, Net.Commerce instances are created, and the sample databases are optionally installed and loaded.

3.11.1 UNIX and Windows NT Net.Commerce configuration

The Net.Commerce configuration manager is a Java Applet that allows you to enter the required configuration parameters and will create, delete, or update an instance.

For each Net.Commerce instance, the configuration will:

- Create instance directories.
- Define TCP/IP ports for the server daemons.
- Set the environment variables that control Net.Commerce in the Net.Commerce configuration files and the Net.Data configuration file.
- Update the Domino Go Web server configuration file (httpd.conf) with URL mappings and protection directives. Update the servlet configuration file (servlet.cnf) with information about the Product Advisor servlets (PRO version only).
- Create the DB2 database for Net.Commerce using a supplied script.
- Enable and start the Net.Commerce daemons.

These steps are described in the *IBM Net.Commerce for Windows NT Installation and Getting Started*, GC09-2626 and *IBM Net.Commerce for AIX Installing and Getting Started*, GC09-2627.

To start the configuration on UNIX or Windows NT:

- UNIX

From the directory /usr/lpp/NetCommerce3/server/bin, run:

```
./start_admin_server
```

(run ./stop_admin_server to stop the server.)

- Windows NT

Select **Start -> Programs -> NetCommerce Version 3.1.2 -> Configuration Manager**.

To perform instance configuration on Windows NT or UNIX, complete the following steps:

1. Access the Net.Commerce configuration Web server at the following URL:
`http://<hostname>:4444/`
2. Enter the user ID `webadmin` and password `webibm`.
3. This will bring up the Net.Commerce Configuration Manager that allows you to:
 - Add new instances

- Delete instances
- View/change instance settings
- Start instances
- Stop instances

See Figure 17.

Net.Commerce CONFIGURATION MANAGER

Use Configuration Manager to change the settings for Net.Commerce components.
You can accept the defaults or make changes to all enabled fields.
Click the tabs to switch components.

Net.Commerce Web Server Database Payment

Instance Name: mser

Communication Port Base: 16550

Number of Server Processes: 2

Server Options:

☐ Enable Server Cache

☒ Use Default Merchant Key

Merchant Key:

Finish Cancel

Figure 17. Windows NT/UNIX Net.Commerce configuration screen

3.11.2 OS/390 Net.Commerce configuration

OS/390 Net.Commerce is configured using an ISPF dialog. This dialog can be accessed from TSO option 6, by running the command `CMNCONF ncddata_file`. `ncdata_file` is the fully-qualified name of the Net.Commerce configuration data file. If you do not specify a name, the configuration program uses `/etc/ncconfig.dat`. Refer to the *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862 guide to configure the correct environment on your OS/390 system. See Figure 18 on page 37.

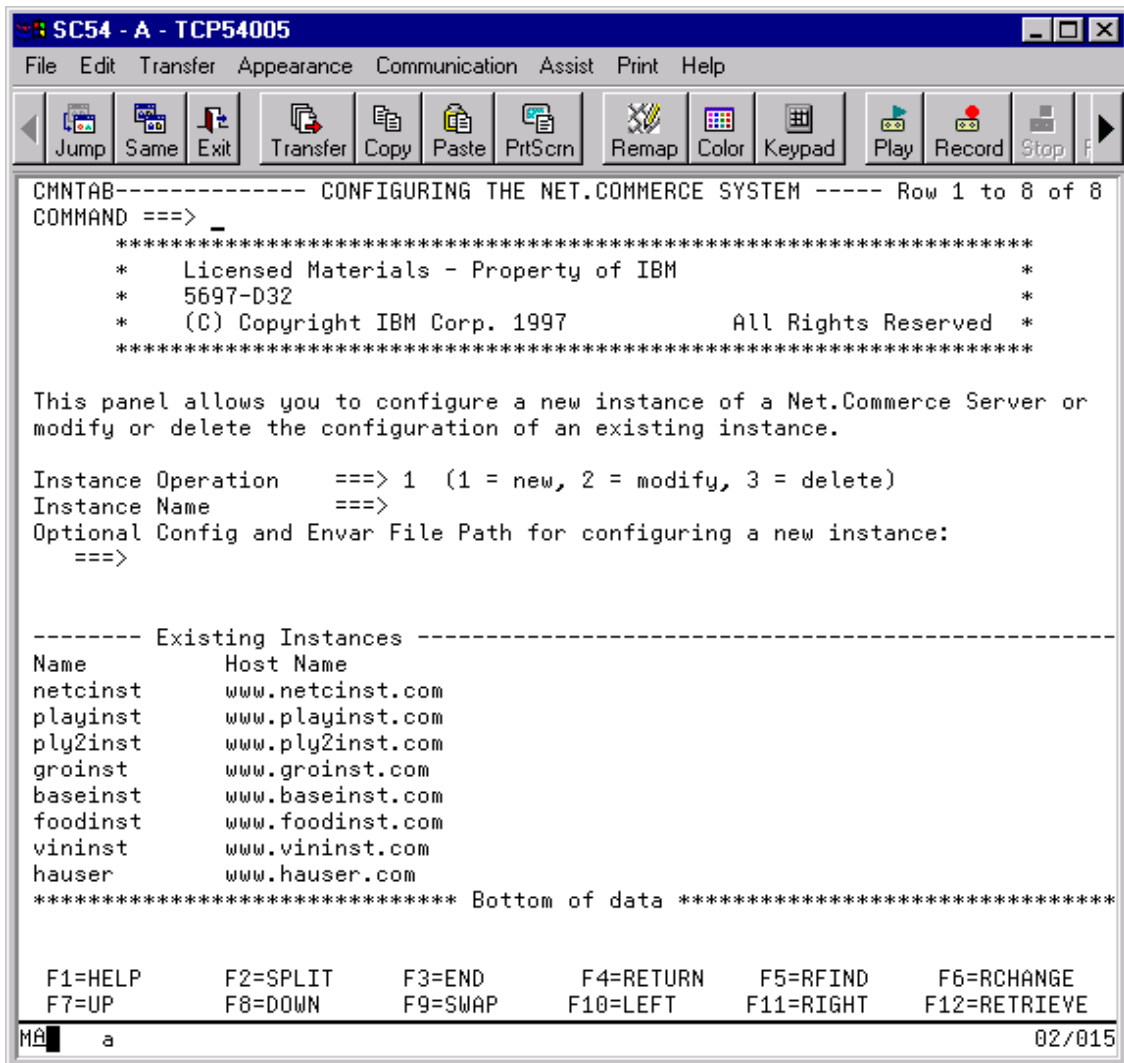


Figure 18. OS/390 CMNCONF screen

3.11.3 Input to CMNCONF

The CMNCONF utility gathers information from a number of different sources:

- The /etc/nccconfig.dat file is used to track all of the instances of Net.Commerce that have already been configured.
- The /etc/services file is used by CMNCONF to see what port ranges have already been reserved.
- The /etc/ncommerce.envvars file is used to provide a set of environment variables for the CMNCONF program itself to use.
- The /usr/lpp/NetCommerce/install/ncommerce.envsamp file will serve as the model input for the environment variables that will be copied to the instance's HTML root directory and the /etc/NC directory when the CMNCONF is finished.
- The /usr/lpp/NetCommerce/html/en_US/db2www.ini file is copied to the instance's document root by CMNCONF and then modified based on the input provided in the fields of the CMNCONF screens.

- Any model configuration file that you point to on the first panel of the CMNCONF utility. Using this feature may save time if you have fields in the configuration that are common across all instances configured.
- All other instance configuration files will be checked to make sure there's no overlap of information between instances. For example, the same owning RACF ID cannot be used in two instances pointing to different database names and the same database name cannot be used in two different instances with different owning RACF IDs.
- The information you supply in the fields of the CMNCONF application.
- The MVS data set hlq.SCMNSAMP holds the SQL script used to create the sample database (if that's the selection you make, as opposed to creating a custom database).
- There's another file that contains SQL script to complete the creation of the database:

```
/usr/lpp/NetCommerce/html/en_US/demomall/data/democom.sql
```
- Finally, the data itself that is loaded into the tables is located in HFS files. Each table is represented by a separate file. The bulk of the data to be loaded is in:

```
/usr/lpp/NetCommerce/html/en_US/demomall/data
```
- The other sample malls (Grocery, Euromall, and Business-to-Business) also have /data directories that appear to offer the incremental differences those malls have over the Demomall.

3.11.4 Output from CMNCONF

A schematic of the output from CMNCONF is shown in Figure 19 on page 39.

.confold file. This is simply the previous version of the file backed up and named with a .confold extension.

These files are known as the Server Controller configuration files.

The contents of the <instance name>.conf file point to the ncommerce.conf file that will be used (reference number 8 in Figure 18 on page 37).

7. A directory will be created off of the /usr/lpp/NetCommerce/html/en_US directory with a name equal to the name of your configured instance. It's here that all the server configuration files are kept. This directory is also known as your Instance HTML root directory, or sometimes the Net.Commerce document root.

Please note that this is the second set of *.conf and *.envvars files for a Net.Commerce setup. The other is located in /etc/NC.

8. The files in this directory are always going to be named db2www.* and ncommerce.*. These are the primary configuration files for the instance.

Please note that the ncommerce.prts file is not actually created by CMNCONF, but rather when the Net.Commerce server is started. We've shown it here just so you won't be surprised if you browse the directory later.

9. A directory will be created off of the /usr/lpp/NetCommerce/instance directory with a name equal to the name of your configured instance. There will be subdirectories under this directory, and it's in here that files created using the Template Designer. In the act of creating the instance, a handful of example files will be copied into these lower subdirectories.

10. Finally, if you choose to create and load a database, the information is added to DB2.

3.11.5 Placing your HTML files in a non-default directory

HTML files are referenced via the Pass directives in the Web server's httpd.conf file. For example, to access the Metropolitan Demomall's front page, you put the following in the browser's location window:

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

By default, there is no directory called /demomall off the root directory, so a Pass directive is added by the CMNCONF utility. The Pass directive added to httpd.conf looks like this:

```
Pass /demomall/* /usr/lpp/NetCommerce/html/en_US/demomall/*
```

With this Pass directive in place, when the Web server receives a URL request with /demomall immediately following the host name, it is resolved to the location of the full path as specified on the Pass directive.

Therefore, to create your own directory for custom HTML files, you would do the following:

1. Create the directory.
2. Add a Pass directive in the Web server httpd.conf file.

Note

Do not put this Pass directive in the directives added by CMNCONF that are located at the top of the httpd.conf file. CMNCONF will overwrite that.

3. Restart the Web server.

You may, if you desire, put custom HTML files in a directory that's created by CMNCONF.

3.11.6 Placing your Net.Data macros in a non-default directory

To achieve this, modify the db2www.ini file, which is in the Net.Commerce instance's html root, and add the directory to the MACRO_PATH directive. The html root is located at:

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

Where <instance_name> is the name of the instance you wish to modify. The db2www.ini file's layout looks like the following:

```
MACRO_PATH /usr/lpp/NetCommerce/instance/<inst_name>/teditor;  
           /usr/lpp/NetCommerce/macro/en_US/ncsample;  
           /usr/lpp/NetCommerce/macro/en_US/demomall;  
           /usr/lpp/NetCommerce/macro/en_US  
INCLUDE_PATH /usr/lpp/NetCommerce/macro/en_US/ncsample;  
            /usr/lpp/NetCommerce/macro/en_US;  
            /usr/lpp/NetCommerce/instance/<inst_name>/teditor;  
            /usr/lpp/NetCommerce/html/en_US;  
ENVIRONMENT (DTW_SQL) dtwsq1 ()  
ENVIRONMENT (DTW_SYS) sysdll ()  
ENVIRONMENT (DTW_SYSTEM) sysdll ()  
ENVIRONMENT (DTW_PERL) perl.dll ()  
ENVIRONMENT (DTW_REXX) rexx.dll ()  
ENVIRONMENT (DTW_FILE) filedll ()  
DB2MSGs NONE  
DTW_REMOVE_WS NO
```

Please note that the MACRO_PATH and INCLUDE_PATH lines actually run on one line, but we broke them across multiple lines here to illustrate it better.

The MACRO_PATH is what Net.Commerce uses (or, more precisely, what Net.Data uses) to search through for macros called. It operates just like any other path directive: The directories are searched in the order of their appearance on the MACRO_PATH directive. So if you want to add a custom directory, you simply add it to the search order. For example:

```
MACRO_PATH /my_custom_directory;  
           /usr/lpp/NetCommerce/instance/<inst_name>/teditor;  
           /usr/lpp/NetCommerce/macro/en_US/ncsample;  
           /usr/lpp/NetCommerce/macro/en_US/demomall;  
           /usr/lpp/NetCommerce/macro/en_US
```

Once added, you need to restart Net.Commerce for the change to take effect.

Note

There's a twist to this for category and product macros you wish to invoke using the CategoryDisplay and ProductDisplay commands. For those two commands only, Net.Commerce assumes the macro will be found in a subdirectory offset of /category and /product off the directories listed on the MACRO_PATH directive in db2www.ini. Therefore, your custom directory for category and product pages (that will be invoked with CategoryDisplay and ProductDisplay) should be subdirectories of /category and /product off of the custom directory named on MACRO_PATH. For example, the directory shown on MACRO_PATH above is /my_custom_directory. Create two subdirectories off that directory, one named /product and one named /category. Leave the MACRO_PATH designation /my_custom_directory. Place your category and product macros in their respective subdirectories. Now when CategoryDisplay or ProductDisplay is run, it'll assume the /category and /product offsets and will find the macros in your custom directory.

Be aware that information you add to db2www.ini will be overwritten whenever you run the Access Control function of CMNCONF (Option 2). You should keep a backup of the file and reapply any changes to db2www.ini any time you run CMNCONF to modify a configuration.

3.11.7 Placing Net.Commerce configuration files into a single directory

You should place all of your Net.Commerce customized files under a single directory. Make that single directory a mount point for a separate HFS from that containing /usr/lpp/NetCommerce. This makes it easier to isolate your custom files from IBM Net.Commerce-supplied files, which may be subject to change through the application of fixes. It also allows you to roll changes from your test system up to your production system by simply cloning the HFS and carrying it over, then unmounting the current production HFS and remounting the new production HFS.

To put all of the configuration files into one directory, follow these steps:

1. Install Net.Commerce. The Net.Commerce HFS will be mounted at the mount point /usr/lpp/NetCommerce (by default) in the root HFS of your system.
2. Determine by what name your instance will be known. We'll refer to it as <inst_name> here.
3. Using OMVS or ISHELL, create a directory called <inst_name> off of the /usr/lpp/NetCommerce/instance directory.

Please note that this directory must have permissions of 755.

4. Allocate a new HFS file and mount it at the /usr/lpp/NetCommerce/instance/<instance name> mount point.

Please note that the allocation of a separate HFS isn't a strict requirement of this process, but the point of this is to have all your custom files in a single HFS that can be easily unmounted and copied.

5. Using OMVS or ISHELL, create a directory called configs off of the /usr/lpp/NetCommerce/instance/<inst_name> directory.

Please note that this directory must have permissions of 755.

Figure 20 shows what you have at this point.

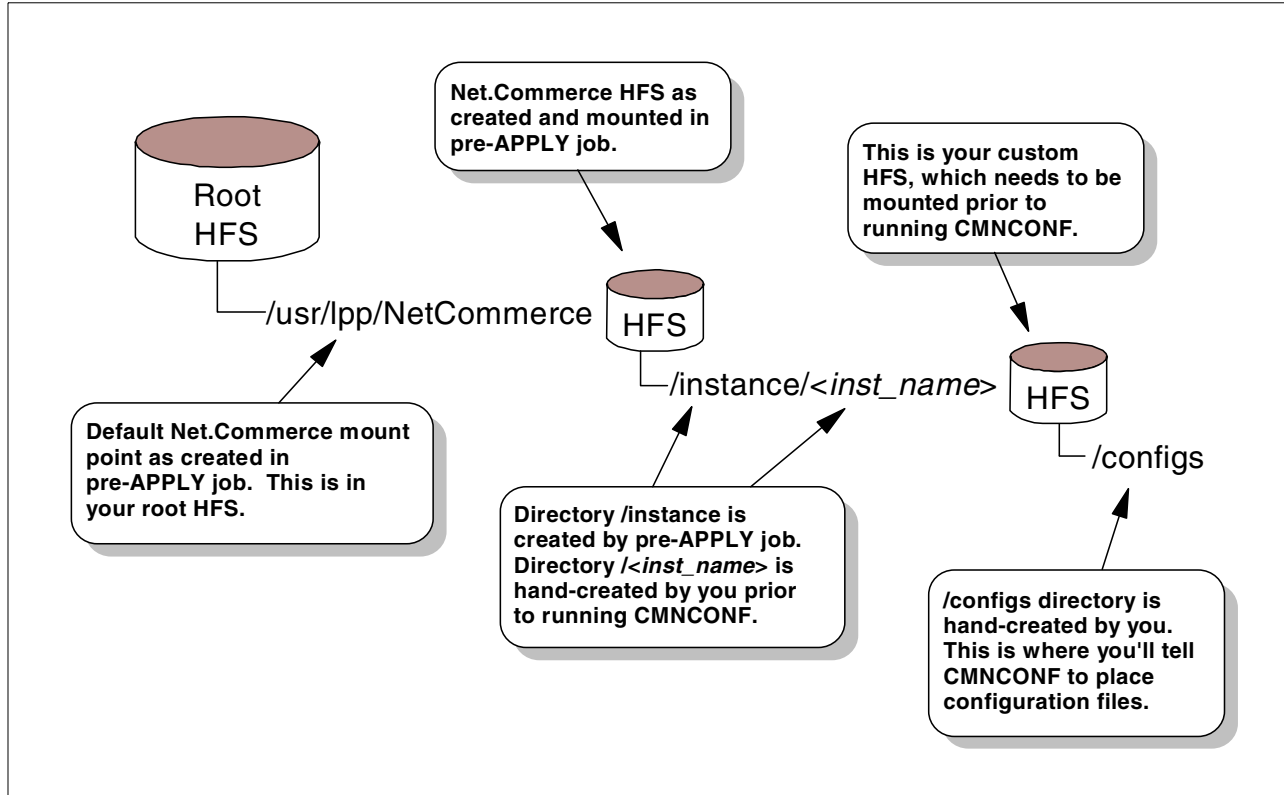


Figure 20. Net.Commerce HFS hierarchy

6. Run CMNCONF to configure your instance. When you get to the System Configuration - Part 1 of 2 panel (under Option 1), type in the following for the Server Controller Conf File field:

```
/usr/lpp/NetCommerce/instance/<inst_name>/configs/<inst_name>.conf
```

The field is long enough to permit a directory and file of this length, but it will wrap.

7. Continue in CMNCONF. On the System Configuration - Part 2 of 2 panel (still under Option 1), enter the following in the HTML Path field:

```
/usr/lpp/NetCommerce/instance/<inst_name>/configs
```

8. Complete the rest of the CMNCONF configuration, including database create and load.

One of the things CMNCONF will do is create a directory called /teditor under the /instance/<inst_name> directory.

9. The sample JCL proc is located in the hlq.SCMNSAMP data set. Its name is CMNMSERV. By default, the PROC statement looks like the following:

```
//CMNMSERV PROC NETCPARM='',
// LE Parm='ENVAR("_CEE_ENVFILE=/etc/NC/-INSTNAME-.envvars")'
```

And a little lower in the proc, the EXEC statement looks like the following:

```
//CMNMSERV EXEC PGM=CMNSRVRC,
// PARM=('&LE Parm/&NETCPARM -i /etc/NC/-INSTNAME-.conf'),
// REGION=0M,TIME=(1440)
```

Note how in both sections LEPARM and PARM point to the default /etc/NC directory. Note also how the value for LEPARM (on the PROC statement) gets placed into the PARM= on EXEC statement (with &LEPARM).

The problem here is that the PARM field has a limit of 100 characters. With a long value for LEPARM of:

```
/usr/lpp/NetCommerce/instance/<inst_name>/configs/<inst_name>.envvars
```

And one of equal length in the PARM field on the EXEC statement, the length quickly exceeds 100 character limit. You'll see an error that looks like this:

```
IEFC642I EXCESSIVE PARAMETER LENGTH ON THE EXEC STATEMENT
```

The way to get around this is to create a symbolic link lower in the HFS structure that points to the <inst_name>.envvars and <inst_name>.conf files deep in the directory structure for your instance.

The structure you are creating through this links is shown in the chart illustrated in Figure 21.

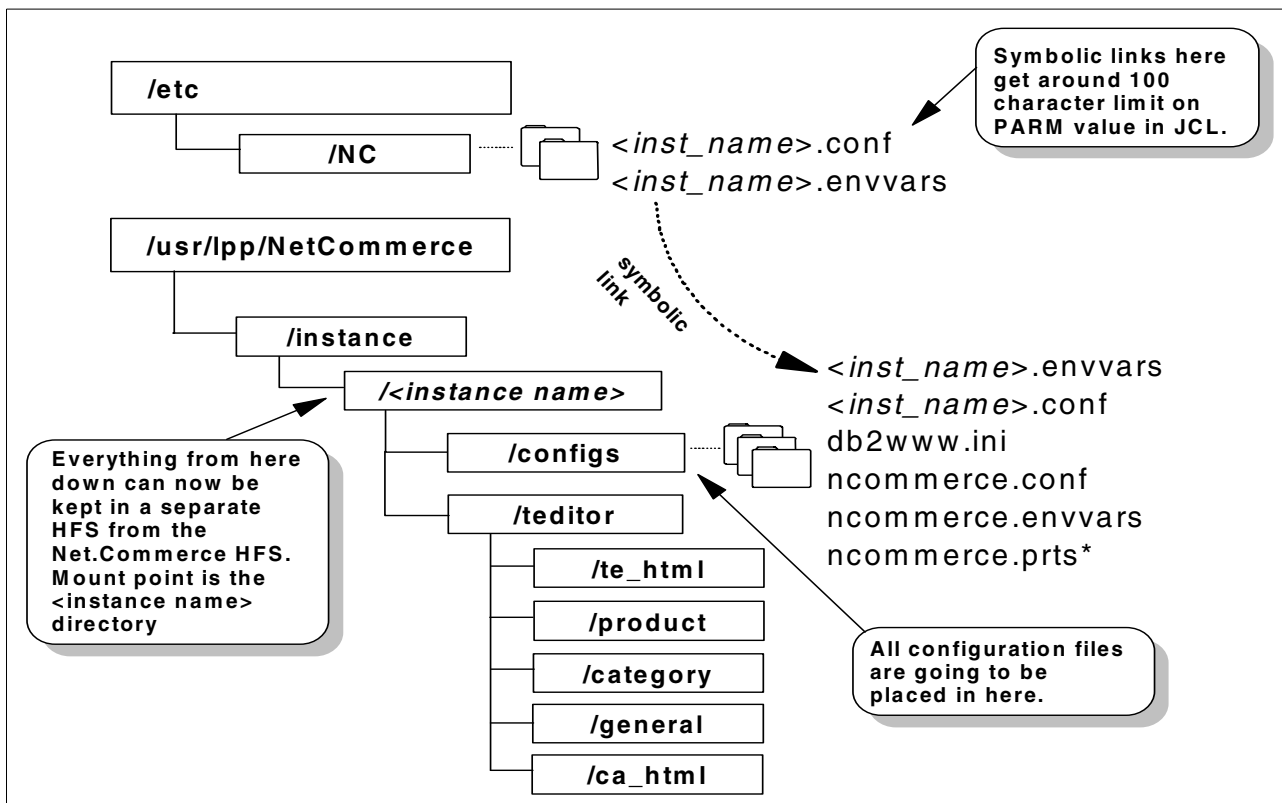


Figure 21. Symbolic links required for config files all go in the same directory

10. Because the sample JCL proc uses /etc/NC, we will as well. Create the directory /etc/NC by hand and provide permission bits of 755. Then create the symbolic links in the /etc/NC directory:

```
ln -s /usr/lpp/NetCommerce/instance/<inst_name>/configs/<inst_name>.envvars
<inst_name>.envvars
ln -s /usr/lpp/NetCommerce/instance/<inst_name>/configs/<inst_name>.conf
<inst_name>.conf
```

11. Now modify the supplied JCL and point to /etc/NC/<inst_name>.envvars and /etc/NC/<inst_name>.conf.

3.11.8 Installation directories

Several directories are created during the installation:

- **Defaults for Windows NT**

Net.Commerce	c:\ibm\NetCommerce3
Lotus Domino Go	c:\ibm\www
DB2	c:\ibm\sqllib

You can select an alternate drive and directory for each component during installation.

- **Defaults for UNIX**

Net.Commerce	/usr/lpp/NetCommerce3
Lotus Domino Go	/usr/lpp/internet
DB2	/usr/lpp/db2_05_00

- **Defaults for OS/390**

Net.Commerce	/usr/lpp/NetCommerce
Web server	/usr/lpp/internet/server_root
DB2 JDBC	/usr/lpp/db2/db2510
DB2	Installed to OS/390 data sets

The directories under the Net.Commerce root are similar on all platforms. Files belonging to a single instance must be kept separate from other instances. Each instance has its own HTML root directory; other files (Net.Data macros, images, and so forth) are located in the Net.Commerce directory structure.

3.12 Net.Commerce operation

The following sections describe the Net.Commerce operations for AIX, Windows NT, and OS/390.

3.12.1 AIX operation

The Web server, Net.Commerce instances, and DB2 are all started and stopped through shell scripts. Refer to the platform documentation for further details.

3.12.2 Windows NT operation

The Web server, Net.Commerce instances, and DB2 are all started and stopped through the Windows NT Services control panel. Refer to the platform documentation for further details.

3.12.3 OS/390 operation

The Web server and Net.Commerce should be controlled as OS/390 started Tasks. DB2 is controlled as an OS/390 subsystem. Your OS/390 systems programmer will have to create started Task job control language (JCL) procedures for the Web server and Net.Commerce.

The startup sequence for the Net.Commerce system components should be as follows:

1. DB2
2. Net.Commerce
3. Web server

3.12.3.1 DB2

DB2 is started as a subsystem of our system with the command:

```
@DSN1 START DB2
```

Where @ is the DB2 subsystem identifier and DSN1 is the DB2 subsystem name.

3.12.3.2 Net.Commerce

A sample JCL start procedure is provided in the data set hlq.SCMNSAMP(CMNMSERV). Copy that member to your system procedures library (proclib) and make the modifications as indicated in the comments of the JCL and then start the Task. The Task can be started from the system display and search facility (SDSF), which can be invoked through a calling ISPF panel, or by going to TSO option 6 and typing SDSF if your system is configured to allow this, as follows:

```
S CMNMSERV
```

The following message in the OS/390 system log indicates that Net.Commerce is up and running:

```
CMN0411I THE NET.COMMERCE SERVER IS READY FOR CONNECTIONS 175
```

Net.Commerce is stopped with the following command:

```
P CMNMSERV
```

It will generally come down gracefully. In the event that it does not, kill it with the command:

```
C CMNMSERV
```

It's recommended that you start Net.Commerce first and then start the Web server. The reason is that the ports file (ncommerce.prt) is not recreated until the Net.Commerce server is started. If you start the Web server first, and there is a port match before the NC server is started, the old ports file will be used (or there will be no ports file at all, resulting in some error message).

3.12.3.3 Web server

The Web server, like Net.Commerce, is started through a procedure, for example:

```
S WEBSRV
```

The following message in the OS/390 system log indicates that the Web server is up and running:

```
IMW3536I SA 436207633 0.0.0.0:80 * * READY
```

You can make changes to the Web server configuration file (httpd.conf by default) and apply those changes dynamically by issuing the following OS/390 console command:

F WEBSRV,APPL=-RESTART

Note the dash (-) between APPL= and RESTART.

The Net.Commerce instance, however, cannot dynamically pick up changes made to the configuration files. Instead, you must stop and start again the Net.Commerce instance.

3.13 Maintaining an OS/390 Net.Commerce site

Typically, the OS/390 system programmer will have separate HFS containers mounted for the files in /usr/lpp/NetCommerce. One HFS container will be the production HFS container, and there will probably be another one mounted so that fixes can be applied to it, for example, /service/usr/lpp/NetCommerce.

Fixes to OS/390 Net.Commerce are delivered in system modification program/extended (SMP/E) format. The fixes are also known as program temporary fixes (PTFs) – they become permanent fixes in reality.

Fixes should first be tested in a test environment. This is typically a separate logical partition (LPAR) from the production LPAR.

When applying a fix, the system programmer should apply it to the /service/usr/lpp/NetCommerce structure by changing the SMP/E DDDEFs or DD statements. To move the fix into production, the following steps should be taken:

1. Backup your current production /usr/lpp/NetCommerce HFS.
2. Backup your current test system /usr/lpp/NetCommerce HFS.
3. Copy the files in /service/usr/lpp/NetCommerce to /usr/lpp/NetCommerce in your test system LPAR (you may need to stop and restart Net.Commerce).
4. Test the fix on the test system LPAR.
5. If the fix works, copy the files to /usr/lpp/NetCommerce on your production system (you may need to stop and restart Net.Commerce).
6. Backup your new production and test /usr/lpp/NetCommerce HFS containers.

Rather than manually copy the contents of the /service/usr/lpp/NetCommerce directory to the production HFS, the systems programmer may prefer to dismount the current production HFS and mount a new HFS that has the fix applied.

If the PTF introduces any problems, the system programmer can quickly reverse the process and remount the old HFS container that does not have the fix applied.

To avoid losing any file updates during the unmount/remount, the system administrator should ensure that those two HFS containers are kept synchronized.

3.14 Summary of the general migration considerations

In this chapter, we discussed the areas that need particular attention when migrating a Net.Commerce application to OS/390, such as ASCII to EBCDIC considerations, Net.Data compatibility, and porting HTML and image files. We also covered the OS/390-specific installation, maintenance, and operation considerations.

Chapter 4. Migrating the database

Net.Commerce uses the DB2 database to access, manipulate, and store its data and logical structures. So, the process of Net.Commerce migration is largely a matter of DB2 migration. While the name DB2 may be common on the different platforms, there are significant differences between the database management system (DBMS) implementations, data structures, and system management requirements on the disparate platforms.

The database schema and information in the tables in a Net.Commerce system can be obtained from the online help information provided with the Net.Commerce product. This online help can be accessed via the browser by specifying the following URL:

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

DBMS and operating system differences, in part, mean Net.Commerce implementations can vary from platform to platform. These concerns must be taken into account when performing a migration across different platforms.

4.1 Overview

Broken down to a logical sequence, there are several steps you must perform in such a migration as you can see in Figure 22 on page 50.

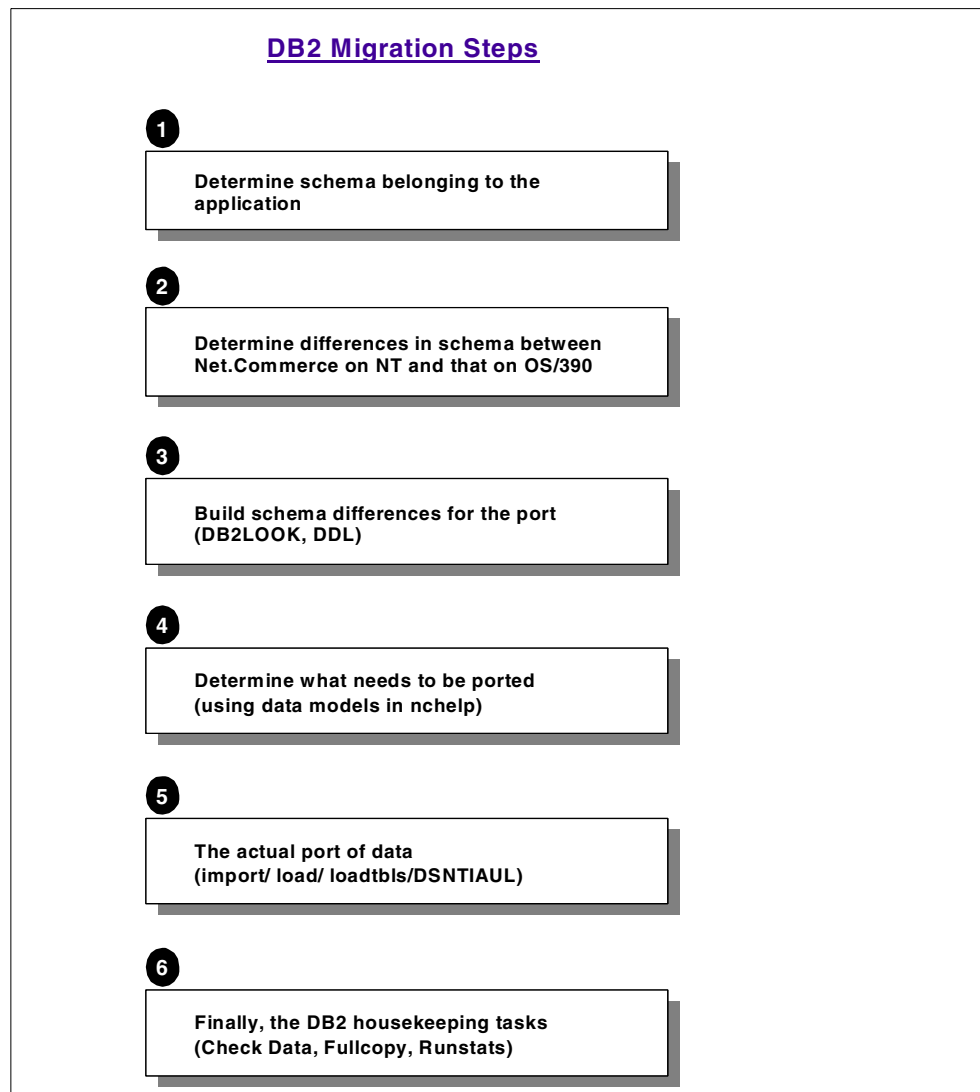


Figure 22. DB2 migration overview

First, you must identify what changes the user application has made on the Net.Commerce system on the existing platform. This would include such things as new tables, views, referential constraints, macros, and functions. You must also identify situations where the user has added columns or constraints to existing Net.Commerce objects. New and added objects will need to be migrated.

Second, you must address schema differences between the Net.Commerce implementations on the different platforms. There are differences in Net.Commerce tables, referential integrity structures, and macros between the Windows NT/UNIX platforms and OS/390. You should verify that all structures referenced or used by the user application being migrated on the source platform exist and are not in conflict with existing structures on the target platform.

After considering what objects must be ported and considering possible conflicts between the operating platforms, you need to create the DB2 data definition language (DDL) required to implement the required migration schema on the OS/390 target. DB2LOOK, supplied by DB2 for Windows NT, is very helpful to

create DDL for the new tables that will be ported to OS/390. With some updates, the output of DB2LOOK can be made DB2 for OS/390 compatible. For user modifications to existing Net.Commerce tables, you will need to either recreate the objects in DB2 for OS/390 (using DB2LOOK output as a source), or create DB2 ALTER statements to migrate schema modifications.

The next step after assuring schema compatibility is to move the application data to the OS/390 target system. There are several ways to accomplish this task. The possibilities include the use of: DB2 for Windows NT/UNIX import/export, DB2 for OS/390 DSNTIAUL, Data Propagator, and Net.Commerce Mass Import. For our example migration, we concentrate on the first two options for moving data.

Finally, after moving the schema and data to the target OS/390 system, you will need to perform some DB2 housekeeping tasks on the migrated system data. Primarily, this includes running the CHECK utility to verify that the data adheres to referential constraints, making copies of the data to ensure future integrity of the data using the COPY utility, reorganizing the data for optimal performance during SQL access using the REORG utility, and collecting new statistics with the RUNSTATS utility so that optimal access path selection can be determined by DB2 at query bind time.

4.1.1 Determine the database objects the user has added/changed

The first thing you must do is identify which DB2 for Windows NT tables will need to be migrated to the DB2 for OS/390 system for your Net.Commerce application. Net.Commerce supplies a base set of objects that are required for its operation, which we will refer to as the BASEMALL. For our particular example, the customer had built on the Net.Commerce DEMOMALL sample. Regardless of what initial sample mall the application to be migrated was built on, you should use that same initial platform as the starting point for your migration on Net.Commerce/390. For our example, all following migration discussion will, therefore, be based on using DEMOMALL as the base of comparison.

Looking at the groups of tables that make up a Net.Commerce application, we decide that the groups of tables that we must concentrate on are:

- Categories, products, and items
- Merchants, stores, shopper groups, and customers
- Orders
- Price discounting
- Shipping
- Shoppers and users
- Tax calculations
- Access control

To complete the port, we would also need information from the following tables:

- MACROS
- KEYS
- OFS
- TASKS
- CMDS
- TASK_MER_OF
- POOLS
- POOL_CMD

- ACC*
- CMD-TASK
- TAXPRCODE

Our initial goal is to identify those tables (and table-related objects) that had been added by the customer in the course of customizing the Net.Commerce for a Windows NT system to their application. We take a generic approach to determine which objects had been added to the system by porting the DB2 for Windows NT SYSIBM.SYSTABLES of an original DEMOMALL instance to the DB2 for the Windows NT system where the user production application existed. A simple SQL outer join was then used to highlight the table/view differences between the two Net.Commerce platforms. This approach contains a lot of power and can be easily adapted to finding differences between any two relational database (RDB) systems with similar catalog structures.

Setup

In concept, SQL join queries are used to compare two Net.Commerce schemas. The SQL joins operate on the DB2 for Windows NT catalog tables of two DB2 instances. To perform the queries, you must either have the IBM Data Joiner program that allows for SQL joins between two separate DB2 systems via DRDA access, or you must copy the relevant DB2 catalog tables into a single DB2 for Windows NT database. With the Data Joiner product, no prior setup is involved other than to enable DRDA connections for the two DB2s. Enabling DRDA between the involved DB2 entities is highly recommended even if the Data Joiner product is not used.

The following steps describe how to setup the DB2 database to run the join queries without the benefit of Data Joiner:

1. Create a new DB2 for Windows NT database instance, which we refer to as MIGRDB.
2. From the DB2 for Windows NT Control Center panel, copy the SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS tables from a standard DEMOMALL DB2 for Windows NT database to the new MIGRDB database. Create the new tables as NTDEMO.SYSTABLES and NTDEMO.SYSCOLUMNS respectively. The assumption is made that sample DEMOMALL and user applications are in separate databases of the same DB2 for Windows NT instance. If this assumption is unworkable in your scenario for some reason, substitute DB2 for Windows NT Export/Import to move the catalog tables.
3. From the DB2 for Windows NT Control Center panel, copy the SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS tables from the DB2 instance that contains the Net.Commerce application to be migrated to the MIGRDB instance. Create the new tables as NTAPP.SYSTABLES and NTAPP.SYSCOLUMNS respectively.

4.1.1.1 Query to locate new tables and views

The following query and sample output is an outer join between the copies of the SYSIBM.SYSTABLES catalog tables of the DB2NT DEMOMALL instance and the DB2 for Windows NT instance containing the user application to be migrated. By using an outer join, tables that exist in one system and not the other can be distinguished. Examining the following SQL query and output, it is evident that five tables exist in the NTAPP DB2/instance that do not exist in the DEMOMALL

instance. Upon investigation of these five tables, it was determined that one, the HISTORY table, was from a rogue application that was not to be ported. The remaining four tables are additions to the base system that need to be recreated and repopulated by the migration.

For the DB2 for Windows NT instance containing the user application, the Net.Commerce creation was accomplished with the user ID SA, while for the DEMOMALL installation, it was accomplished with the user ID TOT14. These SQL predicates will need to be updated for your particular needs.

Note the exclusion of table names beginning with the characters IC, ET, NCET, and NCPAY, as well. During investigation, all such tables and views were found to be representatives of the Payment Server and Product Advisor, which are not part of the described porting procedure.

```
SELECT
    NTDEMO.NAME AS DEMO_TBL, NTAPP.NAME AS APP_TBL,
    NTDEMO.TYPE AS DEMO_TYP, NTAPP.TYPE AS APP_TYP,
    NTDEMO.COLCOUNT AS DEMO_COLS, NTAPP.COLCOUNT AS APP_COLS
FROM (
    (SELECT * FROM NTAPP.SYSTABLES
     WHERE CREATOR='SA'
     AND NAME NOT LIKE 'IC%'
     AND NAME NOT LIKE 'ET%'
     AND NAME NOT LIKE 'NCET%'
     AND NAME NOT LIKE 'NCPAY%')
    ) NTAPP
FULL OUTER JOIN
    (SELECT * FROM NTDEMO.SYSTABLES
     WHERE CREATOR='TOT14'
     AND NAME NOT LIKE 'IC%'
     AND NAME NOT LIKE 'ET%'
     AND NAME NOT LIKE 'NCET%'
     AND NAME NOT LIKE 'NCPAY%')
    ) NTDEMO
ON NTAPP.NAME=NTDEMO.NAME );
```

DEMO_TBL	APP_TBL	DEMO_TYP	APP_TYP	DEMO_COLS	APP_COLS
ACC_CMDGRP	ACC_CMDGRP	T	T	2	2
ACC_GROUP	ACC_GROUP	T	T	4	4
ACC_MODE	ACC_MODE	T	T	5	5
ACC_USRGRP	ACC_USRGRP	T	T	2	2
ACCTRL	ACCTRL	T	T	4	4
API_TASK	API_TASK	V	V	7	7
APIS	APIS	T	T	4	4
BROWSER	BROWSER	T	T	11	11
CACHLOG	CACHLOG	T	T	4	4
CATEGORY	CATEGORY	T	T	12	12
CATEGORY_PUB	CATEGORY_PUB	V	V	12	12
CATESGP	CATESGP	T	T	7	7
CGPRREL	CGPRREL	T	T	4	4
CGRYREL	CGRYREL	T	T	4	4
CMD_PARAM	CMD_PARAM	T	T	3	3
CMD_TASK	CMD_TASK	T	T	2	2
CMDS	CMDS	T	T	9	9
CONSTVD	CONSTVD	V	V	5	5
CONSTVP	CONSTVP	V	V	2	2
DAEMON	DAEMON	T	T	3	3
DISCCALC	DISCCALC	T	T	9	9
DISCCODE	DISCCODE	T	T	5	5
KEYCOL_V	KEYCOL_V	V	V	3	3
KEYS	KEYS	T	T	4	4
MACROS	MACROS	T	T	3	3
MALL	MALL	T	T	12	12
MCUSTINFO	MCUSTINFO	T	T	5	5
MERCHANT	MERCHANT	T	T	29	29
MERCHANTTAX	MERCHANTTAX	T	T	16	16
MSHIPMODE	MSHIPMODE	T	T	7	7
OF_PARAM	OF_PARAM	T	T	3	3
OF_TASK	OF_TASK	T	T	2	2

OFS	OFS	T	T	7	7
ORDER_COMP	ORDER_COMP	V	V	17	17
ORDER_PEND	ORDER_PEND	V	V	17	17
ORDERPAY	ORDERPAY	T	T	17	17
ORDERS	ORDERS	T	T	17	17
ORDPAYMTHD	ORDPAYMTHD	T	T	6	6
PARAMS	PARAMS	T	T	6	6
POOL_CMD	POOL_CMD	T	T	2	2
POOLS	POOLS	T	T	2	2
PRODATR	PRODATR	T	T	5	5
PRODDSTATR	PRODDSTATR	T	T	6	6
PRODPRCS	PRODPRCS	T	T	11	11
PRODSGP	PRODSGP	T	T	7	7
PRODUCT	PRODUCT	T	T	31	31
PRODUCT_PUB	PRODUCT_PUB	V	V	31	31
PROXY	PROXY	T	T	2	2
PRSPCODE	PRSPCODE	T	T	6	6
RATE	RATE	T	T	7	7
SCALE	SCALE	T	T	5	5
SETCURR	SETCURR	T	T	4	4
SETSTATUS	SETSTATUS	T	T	32	32
SG_STORES	SG_STORES	T	T	3	3
SHADDR	SHADDR	T	T	30	30
SHADDR_PERM	SHADDR_PERM	V	V	30	30
SHIPMODE	SHIPMODE	T	T	6	6
SHIPPING	SHIPPING	T	T	14	14
SHIPTO	SHIPTO	T	T	17	17
SHIPTO_COMP	SHIPTO_COMP	V	V	17	17
SHIPTO_PEND	SHIPTO_PEND	V	V	17	17
SHOPDEM	SHOPDEM	T	T	17	17
SHOPGRP	SHOPGRP	T	T	6	6
SHOPPER	SHOPPER	T	T	16	16
SHOPPINGS	SHOPPINGS	T	T	9	9
STATCODE	STATCODE	T	T	4	4
STRCGRY	STRCGRY	T	T	3	3
TAB_V	TAB_V	V	V	2	2
TASK_MER_OF	TASK_MER_OF	T	T	4	4
TASKS	TASKS	T	T	5	5
TAXCGRY	TAXCGRY	T	T	2	2
TAXPRCODE	TAXPRCODE	T	T	4	4
TCOL_V	TCOL_V	V	V	4	4
WTAXMERINFO	WTAXMERINFO	T	T	7	7
-	ACCOUNT_INFO	-	T	-	13
-	HISTORY	-	T	-	3
-	ORDERHISTORY	-	T	-	7
-	ORDERS_MIX	-	T	-	4
-	SHOPLIST	-	T	-	2

79 record(s) selected.

4.1.1.2 Query to locate changed tables and views

The following query is a join of the same copies of the SYSIBM.SYSTABLES catalog tables as in the previous query. In this case, however, the purpose is to determine whether the user has changed any of the columns or referential constraints in the existing Net.Commerce tables. For this particular example, the output shows that no differences were found.

If table changes are found, they must be investigated for porting to the corresponding DB2 for OS/390 tables. This overlaps with the following section where changes between the Net.Commerce schemas are identified. For tables that exist in both platforms, any user modifications to the Net.Commerce for Windows NT base will need to be added to the Net.Commerce for OS/390 base. In cases where a Net.Commerce for Windows NT base table has been modified, and that table does not exist on the OS/390 platform, investigation will be required as to whether the modifications are required for a different table and/or whether the migrated application requires some redesign.

```
SELECT
    NTDEMO.NAME AS TBLNAME,
    NTDEMO.TYPE AS D_TYP, NTAPP.TYPE AS A_TYP,
    NTDEMO.COLCOUNT AS D_COLS, NTAPP.COLCOUNT AS A_COLS,
```

```

        NTDEMO.PARENTS AS D_PAR, NTAPP.PARENTS AS A_PAR ,
        NTDEMO.CHILDREN AS D_CHILD, NTAPP.CHILDREN AS A_CHILD,
        NTDEMO.KEYCOLUMNS AS D_KEYS, NTAPP.KEYCOLUMNS AS A_KEYS
FROM NTAPP.SYSTABLES NTAPP ,
        NTDEMO.SYSTABLES NTDEMO
WHERE NTAPP.NAME=NTDEMO.NAME
        AND NTAPP.CREATOR='SA'
        AND NTDEMO.CREATOR='TOT14'
        AND (
                NTDEMO.COLCOUNT^=NTAPP.COLCOUNT
                OR NTDEMO.PARENTS^=NTAPP.PARENTS
                OR NTDEMO.CHILDREN^=NTAPP.CHILDREN
                OR NTDEMO.KEYCOLUMNS^=NTAPP.KEYCOLUMNS
        );
-----

TBLNAME          D_TYP A_TYP D_COLS A_COLS D_PAR A_PAR D_CHILD A_CHILD D_KEYS A_KEYS
-----

0 record(s) selected.

```

4.1.1.3 Query to locate added columns

The following query is intended to determine which columns the Net.Commerce for Windows NT user application has added to the base Windows NT system. Basically, it is an outer join of the DEMOMALL and user application SYSIBM.SYSTABLES entries that show a difference in the number of columns. As seen in the previous query results that sought out table differences, there is no difference for the sample migration we performed. If new columns exist in tables, this query will show the added columns and their definitions. If found, such columns would need to be investigated and added to the migration target system and/or addressed by application redesign.

```

SELECT
        VALUE(NTDEMO.TBNAME,NTAPP.TBNAME) AS TBNAME,
        VALUE(NTAPP.NAME,NTDEMO.NAME,NTAPP.NAME) AS COLNAME,
        NTDEMO.COLTYPE AS D_TYPE,NTAPP.COLTYPE AS A_TYPE,
        NTDEMO.LENGTH AS D_LGTH,NTAPP.LENGTH AS A_LGTH,
        NTDEMO.NULLS AS D_NULL,NTAPP.NULLS AS A_NULL
FROM (
        (SELECT * FROM NTAPP.SYSCOLUMNS
                WHERE
                        TBcreator='SA'
                AND TBNAME IN
                        (SELECT S.NAME
                                FROM NTAPP.SYSTABLES S,
                                        NTDEMO.SYSTABLES N
                                WHERE S.NAME=N.NAME
                                        AND S.CREATOR='SA'
                                        AND N.CREATOR='TOT14'
                                        AND N.COLCOUNT^=S.COLCOUNT
                        )
        ) NTAPP
FULL OUTER JOIN
        (SELECT * FROM NTDEMO.SYSCOLUMNS
                WHERE
                        TBcreator='TOT14'
                AND TBNAME IN
                        (SELECT N.NAME
                                FROM NTAPP.SYSTABLES S,
                                        NTDEMO.SYSTABLES N
                                WHERE S.NAME=N.NAME
                                        AND S.CREATOR='SA'
                                        AND N.CREATOR='TOT14'
                                        AND N.COLCOUNT^=S.COLCOUNT
                        )
        ) NTDEMO
ON
        NTDEMO.TBNAME=NTAPP.TBNAME
        AND NTDEMO.NAME=NTAPP.NAME );
-----

TBNAME          COLNAME          D_TYPE A_TYPE D_LGTH A_LGTH O_NULL A_NULL
-----

```

0 record(s) selected.

4.1.1.4 Query to locate modified columns

The following query is intended to determine which columns the Net.Commerce for Windows NT user application has modified in the base Windows NT system. This query is a join between Net.Commerce for Windows NT DEMOMALL and user application SYSIBM.SYSCOLUMNS tables, where a difference in column length, type, or null status is sought. As seen in the sample output, our described migration contained no changes to the column definitions of existing tables:

```
SELECT
    NTDEMO.TBNAME AS TBNAME,
    NTDEMO.NAME AS COLNAME,
    NTDEMO.COLTYPE AS D_TYP, NTAPP.COLTYPE AS A_TYP,
    NTDEMO.LENGTH AS D_LGTH, NTAPP.LENGTH AS A_LGTH,
    NTDEMO.NULLS AS D_NULL, NTAPP.NULLS AS A_NULL
FROM NTAPP.SYSCOLUMNS NTAPP,
    NTDEMO.SYSCOLUMNS NTDEMO
WHERE
    NTAPP.TBCREATOR='SA'
AND NTDEMO.TBCREATOR='TOT14'
AND NTDEMO.TBNAME=NTAPP.TBNAME
AND NTDEMO.NAME=NTAPP.NAME
AND (
    NTDEMO.COLTYPE^=NTAPP.COLTYPE
OR NTDEMO.LENGTH^=NTAPP.LENGTH
OR NTDEMO.NULLS^=NTAPP.NULLS
);
```

TBNAME	COLNAME	D_TYP	A_TYP	D_LGTH	A_LGTH	D_NULL	A_NULL

0 record(s) selected.

4.1.2 Differences between NT and OS/390 Net.Commerce tables

In the previous section, we examine the Net.Commerce changes that the user had made. Now, we must also seek out any changes that the Net.Commerce product itself has made between the source and target platforms. You might assume that the system objects used by Net.Commerce for Windows NT and Net.Commerce/390 are the same, but this is not always the case. Other than differences in the operating environment that require different methods of operation, there are also differences in product releases and designs.

There are a significant number of differences between the two platform schemas of Net.Commerce. We found that the most critical in terms of migration are those cases where there are variations in the tables and columns in the OS/390 Net.Commerce schema. The methodology we use to locate Net.Commerce system table discrepancies between the platforms is almost identical to that used to find user implemented changes. The logic is to move the SYSIBM.SYSTABLES and SYSIBM.SYSCOLUMNS tables from the DB2 for Windows NT instance that contains the DEMOMALL sample to a DB2 for OS/390 system containing the DEMOMALL sample and then to use SQL joins to compare the two schemas.

Setup

This setup builds on that used in 4.1.1, "Determine the database objects the user has added/changed" on page 51 where the DEMOMALL and user application Net.Commerce for Windows NT schemas were compared. Basically, the idea is to place copies of the SYSIBM.SYSCOLUMNS and SYSIBM.SYSTABLES DB2 for

Windows NT catalog tables on the OS/390 target system for SQL comparison to the OS/390 schema. Complete the following steps:

1. Enable DRDA connections between the DB2 for OS/390 target system and the DB2 for Windows NT instance created in the previous section as MIGRDB.
2. Install the Net.Commerce DEMOMALL on the DB2 for OS/390 target system. By using a new DB2 unique database name and creator-id, the DEMOMALL addition can be easily separated from other existing DB2 objects via SQL.
3. Extract the DDL for the NTDEMO.SYSTABLES and NTDEMO.SYSCOLUMNS tables. This can be done using the DB2LOOK function supplied with DB2 for Windows NT with the following commands:

```
db2look -d MIGRDB -u userid -t systables -e -o output
db2look -d MIGRDB -u userid -t syscolumns -e -o output
```

4. Edit the resulting Create Table DDL for the NTDEMO.SYSTABLES table by deleting all columns defined with type BLOB, and change the name to NTDEMO.SYSTABLES1. Since the SYSIBM.SYSTABLES table in DB2 for Windows NT contains columns of type BLOB, and DB2 for OS/390 V5 does not support this column type, the DB2 for Windows NT tables must be manipulated to eliminate the offending field types. The columns defined as BLOBs contain no data of interest in the comparison of schemas, so they may be discarded.
5. Execute the edited Create DDL to create the NTDEMO.SYSTABLES1 table.
6. Populate the NTDEMO.SYSTABLES1 table with the data contained in the NTDEMO.SYSTABLES DB2 for Windows NT catalog table. We accomplish this by using a simple SQL Insert operation that ignores the BLOB fields as follows:

```
INSERT INTO NTDEMO.SYSTABLES1
SELECT NAME
CREATOR      , TYPE      , CTIME      , REMARKS
COLCOUNT    , FID       , TID       , CARD
NPAGES       , FPAGES      , OVERFLOW  , PARENTS
CHILDREN     , SELFREFS    , KEYCOLUMNS, KEYOBID
BASE_NAME    , BASE_SCHEMA , TBSPACE   , INDEX_TBSPACE
LONG_TBSPACE , KEYUNIQUE   , CHECKCOUNT, STATS_TIME
DEFINER      , DATA_CAPTURE, STATUS    , CONST_CHECKED
MINPDLENGTH , PMAP_ID     , LOG_ATTRIBUTE, PCTFREE
ROWTYPESCHEMA, ROWTYPEPNAM, APPEND_MODE , PARTITION_MODE
FROM SYSIBM.SYSTABLES
```

- Export the populated NTDEMO.SYSTABLES1 and NTDEMO.SYSCOLUMNS tables either by using the export function available in the DB2 for Windows NT Control Center panel, or using the native DB2 for Windows NT `EXPORT` command. Export the data in IXF format. The native DB2 for Windows NT `EXPORT` command is as follows:

```
export to F:\ntDEMO.systables1.ixf of ixf
select * from ntDEMO.systables1;
export to F:\ntDEMO.sysCOLUMNS.ixf of ixf
select * from ntDEMO.sysCOLUMNS;
```

7. Using the modified DDL for the NTDEMO.SYSTABLES1 and NTDEMO.SYSCOLUMNS tables, create the tables on the DB2 for OS/390 system. For creation on the OS/390 system, we rename the SYSTABLES1 table back to SYSTABLES. The DDL can be executed to the OS/390 DB2 host from the DB2 for Windows NT Command Center following a connect statement. Our DB2 for OS/390 system was defined as having the remote name of DB2P to the DB2 for Windows NT system. The following is an example of the SQL connect statement you must perform prior to executing the Create Table DDL:

```
CONNECT TO DB2P USER HAUSER USING HAUSER;
```

8. Populate the new NTDEMO.SYSTABLES and NTDEMO.SYSCOLUMNS tables on DB2 for OS/390 with the data exported from the same table on DB2 for Windows NT. We perform this by connecting to the remote DB2 for OS/390 system from the Windows NT platform and then using the DB2 for Windows NT `IMPORT` command as follows. Note that this is using the DB2 for Windows NT `IMPORT` function to remotely populate the DB2 for OS/390 tables via DRDA.

```
CONNECT TO DB2P USER HAUSER USING HAUSER;;
import from F:\ntdemo.systables1.ixf of ixf
      insert into ntdemo.systables;
import from F:\ntdemo.syscolumns.ixf of ixf
```

4.1.2.1 Query to locate new tables and views

The primary importance of this is to identify those Net.Commerce for Windows NT tables or views that do not exist on Net.Commerce for OS/390 DEMOMALL. You must investigate whether the user application being migrated references any Windows NT tables and views that are not duplicated on the OS/390 system. Any such references found must be either removed or redesigned to use objects that exist in the OS/390 database to achieve successful migration.

The example query is an outer join between the DB2 for OS/390 SYSIBM.SYSTABLES catalog table and the ported copy of the DB2 for Windows NT SYSIBM.SYSTABLES table containing the Net.Commerce DEMOMALL instance. In the example, a full outer join is used, which shows all DEMOMALL tables and views that exist on one platform exclusively. Examining the output, those tables and views listed in the NT_TBL column represent those that will not exist when building the migrated system on the OS/390 DEMOMALL base.

On the OS/390 system, the 'NETCDBU' user ID was used to create the Net.Commerce DEMOMALL installation in the 'NETCDB' database, while on the Windows NT platform this installation was performed using the 'TOT63' user ID. The query also excludes Payment Server and Product Advisor tables and views, since these are not being ported. These SQL predicates will need to be changed to those required by your particular migration scenario.

```
SELECT
    NT.NAME AS NT_TBL, S390.NAME AS S_TBL,
    NT.TYPE AS NT_TYP, S390.TYPE AS S_TYP,
    NT.COLCOUNT AS NT_COLS, S390.COLCOUNT AS S_COLS
FROM (SELECT * FROM SYSIBM.SYSTABLES
      WHERE CREATOR='NETCDBU'
        AND DBNAME='NETCDB'
        AND NAME NOT LIKE 'IC%'
        AND NAME NOT LIKE 'ET%'
        AND NAME NOT LIKE 'NCET%'
        AND NAME NOT LIKE 'NCPAY%')
    S390
FULL OUTER JOIN
    (SELECT * FROM NTDEMO.SYSTABLES WHERE CREATOR='TOT63'
      AND NAME NOT LIKE 'IC%'
      AND NAME NOT LIKE 'ET%'
      AND NAME NOT LIKE 'NCET%'
      AND NAME NOT LIKE 'NCPAY%')
    NT
ON S390.NAME=NT.NAME
WHERE S390.NAME IS NULL OR NT.NAME IS NULL;
```

NT_TBL	S_TBL	NT_TYP	S_TYP	NT_COLS	S_COLS
CACHLOG	-----	T	----	4	----
CMD_PARAM	-----	T	----	3	----
CMD_TASK	-----	T	----	2	----
CONSTVD	-----	V	----	5	----
CONSTVP	-----	V	----	2	----

```

----- CURRCCACHE ----- T ----- 6
----- CURRCONV ----- T ----- 8
----- CURRCVLIST ----- T ----- 4
----- CURRFCACHE ----- T ----- 11
----- CURRFORMAT ----- T ----- 8
----- CURRLIST ----- T ----- 2
DAEMON ----- T ----- 3 -----
KEYCOL_V ----- V ----- 3 -----
----- NC_CACHE ----- T ----- 4
OF_PARAM ----- T ----- 3 -----
OF_TASK ----- T ----- 2 -----
PARAMS ----- T ----- 6 -----
----- PERFLOG ----- T ----- 9
----- PRODPRC ----- V ----- 13
SETBATCH ----- T ----- 3 -----
SETMERCH ----- T ----- 4 -----
SETPROFILE ----- T ----- 2 -----
SETSTATUS ----- T ----- 32 -----
----- SHIPTOINFO ----- V ----- 10
----- SHOPPING ----- V ----- 10
TAB_V ----- V ----- 2 -----
TCOL_V ----- V ----- 4 -----
----- TEMPADDR ----- T ----- 3
----- TEMPSHOP ----- T ----- 2
----- TMPORDER ----- T ----- 4
----- TOTTABLE ----- V ----- 9
----- USRTRAFFIC ----- T ----- 16
WTAXMERINFO ----- T ----- 7 -----
DSNE610I NUMBER OF ROWS DISPLAYED IS 33

```

4.1.2.2 Query to locate changed tables and views

The next step is to identify those cases where the common Net.Commerce DEMOMALL tables and views have different definitions in terms of the number of columns and referential constraints. Differences identified here must be investigated to determine if the user application requires any adjustment to operate on the different schema on the OS/390 platform.

The sample query used to locate platform variations in the table or view definitions is a join between the DB2 for OS/390 SYSIBM.SYSTABLES table and the ported SYSIBM.SYSTABLES table of the Net.Commerce for Windows NT DEMOMALL installation. Each row of output from this query represents a case where a difference in the number of columns or referential constraints exists in a common DEMOMALL table or view. This query is not definitive, but it does serve as a prompt for additional investigation of cases where differences are noted.

```

-SELECT
    NT.NAME AS TBLNAME,
    NT.TYPE AS N_T, S390.TYPE AS S_T,
    NT.COLCOUNT AS N_COL, S390.COLCOUNT AS S_COL,
    NT.PARENTS AS N_PAR, S390.PARENTS AS S_PAR,
    NT.CHILDREN AS N_CHI, S390.CHILDREN AS S_CHI,
    NT.KEYCOLUMNS AS N_KEY, S390.KEYCOLUMNS AS S_KEY
FROM SYSIBM.SYSTABLES S390,
     NTDEM312.SYSTABLES NT
WHERE S390.NAME=NT.NAME
AND S390.CREATOR='NETCDBU'
AND S390.DBNAME='NETCDB'
AND NT.CREATOR='TOT63'
AND (
    NT.COLCOUNT^=S390.COLCOUNT
    OR NT.PARENTS^=S390.PARENTS
    OR NT.CHILDREN^=S390.CHILDREN
    OR NT.KEYCOLUMNS^=S390.KEYCOLUMNS
)
;

```

TBLNAME	N_T	S_T	N_COL	S_COL	N_PAR	S_PAR	N_CHI	S_CHI	N_KEY	S_KEY
CATEGORY	T	T	12	13	1	1	6	6	1	1
CATEGORY_PUB	V	V	12	13	0	0	0	0	0	0
MERCHANT	T	T	29	30	1	1	33	35	1	1
SHOPPER	T	T	16	17	0	0	8	9	1	1

SHOPGRP	T	T	6	7	1	1	4	4	1	1
SHIPMODE	T	T	6	12	0	0	1	1	1	1
MSHIPMODE	T	T	7	8	2	2	1	1	1	1
SHIPPING	T	T	14	15	3	3	0	0	1	1
PRODUCT	T	T	31	32	3	3	6	6	1	1
PRODUCT_PUB	V	V	31	32	0	0	0	0	0	0
PRODPRCS	T	T	11	12	3	3	0	0	1	1
PRODATR	T	T	5	7	2	2	0	0	0	0
PRODDSTATR	T	T	6	7	2	2	0	0	0	0
PRODSGP	T	T	7	8	3	3	0	0	0	0
ORDERS	T	T	17	17	3	3	4	3	1	1
SHIPTO	T	T	17	21	4	4	0	0	1	1
SHIPTO_COMP	V	V	17	21	0	0	0	0	0	0
TASKS	T	T	5	5	0	0	6	4	1	1
CATESGP	T	T	7	8	3	3	0	0	0	0
DISCCALC	T	T	9	10	4	4	0	0	0	0
SETCURR	T	T	4	4	0	0	0	6	1	1
CMDS	T	T	9	9	0	0	5	3	1	1
OFS	T	T	7	7	0	0	4	2	1	1
ACC_USRGRP	T	T	2	3	2	2	0	0	0	0
SHIPTO_PEND	V	V	17	21	0	0	0	0	0	0

DSNE610I NUMBER OF ROWS DISPLAYED IS 25

4.1.2.3 Query to locate added columns

The purpose of this exercise is to identify those cases where columns have been added, subtracted, or renamed between the DEMOMALL OS/390 and Windows NT schemas. The important case to consider is where an Windows NT column has been deleted or renamed on the OS/390 DEMOMALL schema. If the user application on the Windows NT platform were to reference such fields, the application would need to be modified for migration.

The sample query provided is an outer join between the SYSIBM.SYSCOLUMNS tables of the OS/390 and Windows NT DEMOMALL installations. On a table-by-table basis, any column occurring on one platform and not the other is listed. As seen in the output, all listed columns are from the OS/390 system, as would be expected from the results of the previous query to locate changes in tables or views.

For this example, there is no relevance to the migration since all fields in the common DEMOMALL Windows NT installation exist on the OS/390 platform. In the future, however, this result could not be assumed. You should investigate the possibility of fields existing on the Net.Commerce for Windows NT platform that do not exist on the corresponding OS/390 platform.

```

SELECT
    VALUE(NT.TBNAME,S390.TBNAME) AS TBNAME,
    VALUE(S390.NAME,NT.NAME,S390.NAME) AS COLNAME,
    NT.COLTYPE AS N_TYPE,S390.COLTYPE AS S_TYPE,
    NT.LENGTH AS N_LGTH,S390.LENGTH AS S_LGTH,
    NT.NULLS AS N_NULL,S390.NULLS AS S_NULL
FROM (SELECT * FROM SYSIBM.SYSCOLUMNS
WHERE
    TBCREATOR='NETCDBU'
    AND TBNAME IN
        (SELECT S.NAME
        FROM SYSIBM.SYSTABLES S,
        NTDEMO.SYSTABLES N
        WHERE S.NAME=N.NAME
        AND S.CREATOR='NETCDBU'
        AND S.DBNAME='NETCDB'
        AND N.CREATOR='TOT63'
        AND N.COLCOUNT^=S.COLCOUNT
        )
    ) S390
FULL OUTER JOIN
    (SELECT * FROM NTDEMO.SYSCOLUMNS
    WHERE
        TBCREATOR='TOT63'
        AND TBNAME IN

```



```

        (SELECT N.NAME
        FROM SYSIBM.SYSTABLES S,
             NTDEMO.SYSTABLES N
        WHERE S.NAME=N.NAME
             AND S.CREATOR='NETCDBU'
             AND S.DBNAME='NETCDB'
             AND N.CREATOR='TOT63'
             AND N.COLCOUNT^=S.COLCOUNT
        )
    ) NT
ON
    NT.TBNAME=S390.TBNAME
    AND NT.NAME=S390.NAME
WHERE NT.NAME IS NULL OR S390.NAME IS NULL
;

```

TBNAME	COLNAME	N_TYPE	S_TYPE	N_LGTH	S_LGTH	N_NULL	S_NULL
ACC_USRGRP	MER_RFNBR	-----	INTEGER	-----	4	-----	Y
CATEGORY	CGOID	-----	CHAR	-----	36	-----	Y
CATEGORY_PUB	CGOID	-----	CHAR	-----	36	-----	Y
CATESGP	CSTOID	-----	CHAR	-----	36	-----	Y
DISCCALC	DCCSCLCURR	-----	CHAR	-----	3	-----	Y
MERCHANT	MERIOD	-----	CHAR	-----	36	-----	Y
MSHIPMODE	MMTRKSPNBR	-----	VARCHAR	-----	64	-----	Y
PRODATR	PAOID	-----	CHAR	-----	36	-----	Y
PRODATR	PATYPE	-----	INTEGER	-----	4	-----	Y
PRODDSTATR	PDROID	-----	CHAR	-----	36	-----	Y
PRODPRCS	PPOID	-----	CHAR	-----	36	-----	Y
PRODSGP	PSTOID	-----	CHAR	-----	36	-----	Y
PRODUCT	PROID	-----	CHAR	-----	36	-----	Y
PRODUCT_PUB	PROID	-----	CHAR	-----	36	-----	Y
SHIPMODE	SMTRKICON	-----	VARCHAR	-----	64	-----	Y
SHIPMODE	SMTRKNAME	-----	VARCHAR	-----	64	-----	Y
SHIPMODE	SMTRKSH	-----	VARCHAR	-----	64	-----	Y
SHIPMODE	SMTRKSP	-----	INTEGER	-----	4	-----	Y
SHIPMODE	SMTRKTYPE	-----	CHAR	-----	8	-----	Y
SHIPMODE	SMTRKURL	-----	VARCHAR	-----	64	-----	Y
SHIPPING	SPCURR	-----	CHAR	-----	3	-----	Y
SHIPTO	STBASECURR	-----	CHAR	-----	3	-----	Y
SHIPTO	STBASEPRICE	-----	DECIMAL	-----	15	-----	Y
SHIPTO	STTRKDATE	-----	TIMESTAMP	-----	10	-----	Y
SHIPTO	STTRKNBR	-----	VARCHAR	-----	64	-----	Y
SHIPTO_COMP	STBASECURR	-----	CHAR	-----	3	-----	Y
SHIPTO_COMP	STBASEPRICE	-----	DECIMAL	-----	15	-----	Y
SHIPTO_COMP	STTRKDATE	-----	TIMESTAMP	-----	10	-----	Y
SHIPTO_COMP	STTRKNBR	-----	VARCHAR	-----	64	-----	Y
SHIPTO_PEND	STBASECURR	-----	CHAR	-----	3	-----	Y
SHIPTO_PEND	STBASEPRICE	-----	DECIMAL	-----	15	-----	Y
SHIPTO_PEND	STTRKDATE	-----	TIMESTAMP	-----	10	-----	Y
SHIPTO_PEND	STTRKNBR	-----	VARCHAR	-----	64	-----	Y
SHOPGRP	SGOID	-----	CHAR	-----	36	-----	Y
SHOPPER	SHPPREFERREDCURR	-----	CHAR	-----	3	-----	Y

DSNE610I NUMBER OF ROWS DISPLAYED IS 35
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100

4.1.2.4 Query to locate modified columns

The following query is used to determine whether there have been any definition changes for common columns between the Windows NT and OS/390 DEMOMALL installations of Net.Commerce. Differences in definition for a column could impact the migration if the user application references any such columns. Differences may also be relevant to the success of data migration.

In particular, what is being sought out are cases where column type, length, or nullability status have changed between the two Net.Commerce DEMOMALL implementations. The sample query searches for such differences by executing a join between the SYSIBM.SYSCOLUMNS entries of the Windows NT and OS/390 platforms. From the sample output, it is evident that there is some variation in column definitions between the two platforms that may need to be addressed for a migration.

Those columns defined as type LONGVAR make up a sizeable portion of the query results due to a difference in the implicit lengths. LONGVAR is not defined as having any particular length in reality, and is limited by the database manager to what can be fit into the maximum page size. Note that on the OS/390 system that a 4 K page size is used for the Net.Commerce tablespaces, so it is possible that moving data from Windows NT to OS/390 will result in truncation of LONGVAR data. In cases where this problem is found, placing the offending tables into tablespaces defined with 32 K pages should be performed.

```

SELECT
    NT.TBNAME AS TBNAME,
    NT.NAME AS COLNAME,
    NT.COLTYPE AS NT_TYP,S390.COLTYPE AS S_TYP,
    NT.LENGTH AS NT_LGTH,S390.LENGTH AS S_LGTH,
    NT.NULLS AS NT_NULL, S390.NULLS AS S_NULL
FROM SYSIBM.SYSCOLUMNS S390,
     NTDEMO.SYSCOLUMNS NT
WHERE
    S390.TBCREATOR='NETCDBU'
    AND NT.TBCREATOR='TOT63'
    AND NT.TBNAME=S390.TBNAME
    AND NT.NAME=S390.NAME
    AND (
        NT.COLTYPE^=S390.COLTYPE
        OR NT.LENGTH^=S390.LENGTH
        OR NT.NULLS^=S390.NULLS
    )
;

```

TBNAME	COLNAME	NT_TYP	S_TYP	NT_LGTH	S_LGTH	NT_NULL	S_NULL
APIS	APIDLLNAME	CHAR	VARCHAR	254	254	Y	Y
APIS	APIFUNCNAME	CHAR	VARCHAR	254	254	Y	Y
CATEGORY	CGLDESC	LONGVAR	LONGVAR	32700	2252	Y	Y
CATEGORY_PUB	CGLDESC	LONGVAR	VARCHAR	32700	2252	Y	Y
CATESGP	CDESC	LONGVAR	LONGVAR	32700	3226	Y	Y
CGPRREL	CPSEQNBR	DOUBLE	FLOAT	8	8	Y	Y
CGRYREL	CRSEQNBR	DOUBLE	FLOAT	8	8	Y	Y
ICPRODPRICE	PLCGNBR	INTEGER	INTEGER	4	4	N	Y
ICPRODPRICE	PLMENBR	INTEGER	INTEGER	4	4	N	Y
ICPRODPRICE	PLPRRFNBR	INTEGER	INTEGER	4	4	N	Y
MACROS	MAFILENAME	CHAR	VARCHAR	254	254	Y	Y
MALL	MHTHEAD	CHAR	CHAR	254	253	Y	Y
PRODDSTATR	PDSEQNBR	DOUBLE	FLOAT	8	8	Y	Y
PRODSGP	PSDESC	LONGVAR	LONGVAR	32700	3226	Y	Y
PRODUCT	PRLDESC1	LONGVAR	LONGVAR	32700	718	Y	Y
PRODUCT	PRLDESC2	LONGVAR	LONGVAR	32700	718	Y	Y
PRODUCT	PRLDESC3	LONGVAR	LONGVAR	32700	718	Y	Y
PRODUCT_PUB	PRLDESC1	LONGVAR	VARCHAR	32700	718	Y	Y
PRODUCT_PUB	PRLDESC2	LONGVAR	VARCHAR	32700	718	Y	Y
PRODUCT_PUB	PRLDESC3	LONGVAR	VARCHAR	32700	718	Y	Y
STATCODE	STANAME	CHAR	CHAR	40	40	Y	N
TAB_V	REMARKS	VARCHAR	VARCHAR	254	254	Y	N
TASKS	TKCOMMENT	CHAR	VARCHAR	254	254	Y	Y
TAXPRCODE	TPCDESC	LONGVAR	LONGVAR	32700	3984	Y	Y
TCOL_V	REMARKS	VARCHAR	VARCHAR	254	254	Y	N

DSNE610I NUMBER OF ROWS DISPLAYED IS 25

4.1.3 Common tables of Windows NT and OS/390 Net.Commerce

The purpose here is to generate a list of the tables that are common between the Windows NT and OS/390 Net.Commerce DEMOMALL installations. The table members of the resulting list, combined with the user-added tables as determined earlier in 4.1.2.1, "Query to locate new tables and views" on page 58, comprise all the tables whose data must be ported to the OS/390 target platform. The required query is a simple join between the Windows NT and OS/390 versions of the DB2 SYSIBM.SYSTABLES catalog table:

```

SELECT
    Windows NT.CREATOR AS NT_CREATOR,
    NT.NAME AS NT_TBLNAME

```

```

FROM SYSIBM.SYSTABLES S390 ,
      NTDEMO.SYSTABLES NT
WHERE S390.NAME=NT.NAME
      AND S390.CREATOR='NETCDBU'
      AND S390.DBNAME='NETCDB'
      AND NT.CREATOR='TOT63'
      AND NT.NAME NOT LIKE 'IC%'
      AND NT.NAME NOT LIKE 'ET%'
      AND NT.NAME NOT LIKE 'NCET%'
      AND NT.NAME NOT LIKE 'NCPAY%'
;
-----+-----+-----+-----+-----+-----+
NT_CREATOR  NT_TBLNAME
-----+-----+-----+-----+-----+
TOT63      CATEGORY
TOT63      CATEGORY_PUB
TOT63      CGRYREL
TOT63      MALL
TOT63      BROWSER
TOT63      STRCGRY
TOT63      MERCHANT
TOT63      SHOPPER
TOT63      SHOPGRP
TOT63      MCUSTINFO
TOT63      SHOPDEM
TOT63      SHADDR
TOT63      SHADDR_PERM
TOT63      ACCTRL
TOT63      TAXCGRY
TOT63      TAXPRCODE
TOT63      SHIPMODE
TOT63      MSHIPMODE
TOT63      PRSPCODE
TOT63      SHIPPING
TOT63      PRODUCT
TOT63      PRODUCT_PUB
TOT63      CGPRREL
TOT63      PRODPRCS
TOT63      PRODATR
TOT63      PRODDSTATR
TOT63      PRODSGP
TOT63      SHOPPINGS
TOT63      ORDERS
TOT63      SHIPTO
TOT63      ORDERPAY
TOT63      ORDPAYMTHD
TOT63      SHIPTO_COMP
TOT63      STATCODE
TOT63      TASKS
TOT63      API_TASK
TOT63      MACROS
TOT63      APIS
TOT63      KEYS
TOT63      CATESGP
TOT63      SCALE
TOT63      RATE
TOT63      DISCCODE
TOT63      DISCCALC
TOT63      MERCHANTTAX
TOT63      SETCURR
TOT63      POOLS
TOT63      CMDS
TOT63      OFS
TOT63      POOL_CMD
TOT63      TASK_MER_OF
TOT63      PROXY
TOT63      ACC_GROUP
TOT63      ACC_CMDGRP
TOT63      ACC_USRGRP
TOT63      ACC_MODE
TOT63      SG_STORES
TOT63      ORDER_COMP
TOT63      ORDER_PEND
TOT63      SHIPTO_PEND
DSNE610I NUMBER OF ROWS DISPLAYED IS 60

```

4.1.4 Determine high-level DB2 for OS/390 database design

Before moving forward, at this point, you should make some decisions that will affect the performance, monitoring capability, maintenance scenarios, ease-of-tuning, and manageability of your resulting OS/390 Net.Commerce system. This can seem a daunting task and certainly would be if carried to extreme. Therefore, it is best to address the basics, move through migration, and then readdress any further improvements as appropriate for the fully migrated system.

4.1.4.1 Stogroups

You must determine what set (or sets) of physical disk volumes you will store your Net.Commerce database objects in. Within DB2 for OS/390, there are two methods used to define the physical storage location of objects: the user-defined and stogroup methods. With the user-defined method, you must create and manage the VSAM linear data sets yourself. In earlier days, there were some benefits of user-defined data sets, but today, the vast majority of users use stogroup definitions that allow DB2 to control the management and placement of data sets.

It is recommended that you create at least one stogroup that will contain the volumes that will be exclusively used to store your database objects. Beyond this minimum recommendation, you can also decide to define multiple stogroups with which you can implement schemes by which you can assure the segregation of data sets across volumes. An example would be to create different stogroups for tables and indexes, whereby indexes are never resident on the same volume as the tablespace it references. The benefit of such a scheme would be to reduce I/O contention during index access of a table.

4.1.4.2 Tablespaces

A tablespace in DB2 for OS/390 is a construct to map logical tables to physical VSAM data sets. A tablespace can contain one or more tables, while a table must be defined to be in a single tablespace. What you must decide is how you wish to spread your tables across physical tablespaces. The extremes possible are that you could have one tablespace within which all your tables reside, or you could have each table dedicated to its own private tablespace.

There are a fair number of factors that can be considered when deciding how to define tablespaces for tables. The most notable are table size and how the table is accessed by the application. Large tables should be provided a private tablespace, and very large tables should be provided a partitioned tablespace. Small tables can be defined to share a common tablespace, or each can have its own private tablespace. When multiple tables will share a tablespace, the tablespace should be defined as segmented.

For our particular migration, we chose to create each table in its own private tablespace. This allows for the maximum amount of flexibility in the future, although it also means you will have the largest number of data sets to define and manage.

Another factor to consider in your DB2 design is whether you will require tables having rows longer than 4 K. The default for tablespaces is that they are created in the BP0 buffer pool, which is a 4 K page size buffer pool. In DB2 for OS/390, a table row cannot span pages, and so, the maximum record length is bounded by

page size. After allowing for page and record headers, the largest row that will fit into a 4 K page is 4056 bytes. If you require longer rows, you will need to create the corresponding tablespace(s) in 32 K buffer pools, where a maximum record length of 32714 bytes is possible. Note that in DB2 for OS/390 Version 6 that, in addition to 4 K and 32 K page sizes, 8 K and 16 K page sizes are also supported.

4.1.4.3 Buffer pools

A final consideration is determining your DB2 buffer pool usage. The DB2 accounting and statistics traces accumulate statistics for each buffer pool separately. Significant tuning can also be done for buffer pools for such things as memory usage, write thresholds, sequential pre-fetch, hiperpool usage, and so on. Therefore, from a performance monitoring and tuning standpoint, you may wish to segregate objects into different buffer pools. A very common methodology is to divide your database objects into four buffer pools: read-only tablespaces, read-only indexes, read-write tablespaces, and read-write indexes occupying different buffer pools. This method allows for significant tuning, diagnostic, and monitoring capabilities.

4.1.5 Data definition language

The physical implementation of database objects is quite different between database systems. There are several reasons for the variation of data definition language (DDL) between database systems. The most glaring is due to differences between different operating environments, and the fact that the database system must be able to create objects within the usage and management capabilities of its host operating system. Another reason for DDL variance is simply due to the fact that DDL has not been widely standardized in the database industry. Because of operating system differences, different supported functionality of the database management systems (DBMS), and different approaches used to implement the different DBMSs, the goal of standardized DDL is not likely to be reached.

Therefore, the SQL DDL required to build the objects used by Net.Commerce must be customized to the specific database system that is the migration target. In this instance, the DDL required for the Net.Commerce on an OS/390 system must be ported and/or derived from the existing Windows NT implementation. The actual mechanics required to translate the DB2 objects from one DB2 system to another will vary from system to system because of differences in user Net.Commerce implementations. And so, the methods documented for this specific Net.Commerce Windows NT-to-OS/390 migration may not be correct for all migrations cases.

As previously discussed, there are two categories of objects of interest in this porting scenario: those provided by the Net.Commerce product itself and those implemented by the customer for their specific application. The goal within this section is to extract those objects that have been modified or added by the customer in support of the application being migrated and to recreate only those needed objects on the target DB2 system.

Be aware that there is certainly more than one way to do this task. We have taken one path, but others surely exist. The methodology you use for your particular migration may well be different due to differences in Net.Commerce applications, the availability of third-party tools, and/or different levels of system expertise within the porting team. We have take a bare-bones approach which does not

require the purchase of any additional tools for the migration process, although it does require a good of knowledge of the DB2 for OS/390 database.

4.1.5.1 Obtain Windows NT DDL (historic file or DB2LOOK)

First, you must collect the DDL required for the user added or modified Net.Commerce Windows NT user objects. As noted previously, four new tables were located in the user application being ported. Note that in addition to new or modified tables, you should also evaluate whether you have new or modified indexes, views, or referential constraints to extract as well.

To extract the DDL from the DB2 for Windows NT, there is a utility called DB2LOOK that is provided with DB2 UDB. It was simple enough to execute DB2LOOK to extract all the DDL representing the Net.Commerce DB2 objects.

DB2LOOK is a DOS executable present in the c:\ibm\sqllib\misc directory (if you have installed DB2 on C: on Windows NT). It should be executed with the following options:

```
db2look -d DBname -u Creator -t Tname -e -o Fname
```

The syntax of DB2LOOK is shown in the capture illustrated in Figure 23. All objects connected to the DB2 table specified in the DB2LOOK syntax will be included in the generated DDL including indexes, views, and referential constraints.

```
C:\IBM\SQLLIB\misc>db2look
db2look Version 5.0

Syntax: db2look -d DBname [-u Creator] [-s] [-g] [-a] [-t Tname1 Tname2...TnameN]
        [-p] [-o Fname]
        db2look -d DBname [-u Creator] [-a] [-e] [-t Tname1 Tname2...TnameN]
        [-m] [-c] [-r] [-o Fname]
        db2look [-h]

        -d: Database name. *MUST* be specified

        -a: generate stats for all creators
        -c: do not generate commit statement for mimic
        -e: extract DDL file needed to duplicate database
        -g: use graph to show fetch page pairs for indices
        -h: more detailed help message
        -m: run the program in mimic mode
        -o: output filename
        -p: use plain text format
        -r: do not generate runstats statement for mimic
        -s: generate postscript file
        -t: generate stats for the specified tables
        -u: creator ID. If -u and -a are both not specified then
            Environment variable USER will be used
```

Figure 23. DB2LOOK syntax

4.1.5.2 Create stogroup(s) and tablespaces

Create the DDL to create the DB2 for OS/390 stogroups, databases, and tablespaces you have determined are needed for the new objects being ported to the Net.Commerce/390 platform.

For assistance in sizing these objects, you may find the DB2 Estimator product helpful. With this product, you can estimate data set primary and secondary quantities for tablespaces and indexes. This information is required for the CREATE TABLESPACE and CREATE INDEX DDL statements.

For our particular migration, the tables were quite small, so the DB2 default sizings were sufficient. For other cases, however, if the tables are large, you should estimate the final sizes using DB2 Estimator. The simplest way to input all required information into the Estimator product is use the Import Table function of Estimator. After modifying the CREATE TABLE and CREATE INDEX statements in the following step, the resulting DDL can be used in the Estimator Import Table function to input the definitions. The table row counts can then be inserted into Estimator for each table in the Table Definition panel, and, finally, the View Space Requirements function of Estimator can be used to divulge the space estimates. These sizing estimates can then be placed into the preliminary CREATE TABLESPACE and CREATE INDEX statements, resulting in the final DDL.

At this stage, you may also wish to evaluate whether you will require any table row lengths longer than 4056 bytes in length. From our own experience with Net.Commerce migration, we found that this can be an issue for the common Net.Commerce tables that contain columns defined as LONGVAR. In our Windows NT environment, a LONGVAR was capable of containing approximately 32000 bytes. In migrating to the OS/390 platform, however, the default page size in DB2 will limit the record length to no more than 4056 bytes. Since the LONGVAR is of variable length, you must do some analysis work to determine whether you can or cannot port your data into 4 K pages on DB2 for OS/390. To obtain longer row lengths, you will need to create tablespaces that use the 32 K page size buffer pools. A longer row size is not a detriment, so we chose to define tablespaces as having 32 K pages for all tables (user and Net.Commerce) where there was even the potential of exceeding the 4056 byte maximum row length of 4 K pages.

4.1.5.3 Translate the table and index DDL

There is a fair amount of tedium involved to translate from DB2 for Windows NT to DB2 for OS/390 DDL syntax. None of it is very difficult, but it may be time consuming and will likely require multiple execute/debug/fix/execute recursions. Our experience with this process while porting the sample follows.

CREATE TABLE DDL was fairly simple to migrate to DB2 for OS/390. Changing the table creator name and tablespace name was sufficient. CREATE INDEX DDL was similarly simple although it also requires data set primary and secondary extent information to be added if you can not use DB2 defaults.

Referential constraints were unloaded by DB2LOOK in the form of ALTER TABLE statements. The syntax for defining referential constraints is quite different between DB2 for Windows NT and DB2 for OS/390. Other than syntax translation, you may also find that you must add new CREATE INDEX statements to ensure the uniqueness of primary key definitions in DB2 for OS/390. This is because under DB2 for Windows NT, the creation of a primary key does not require the existence of a unique index for that key, while under DB2 for OS/390 it does.

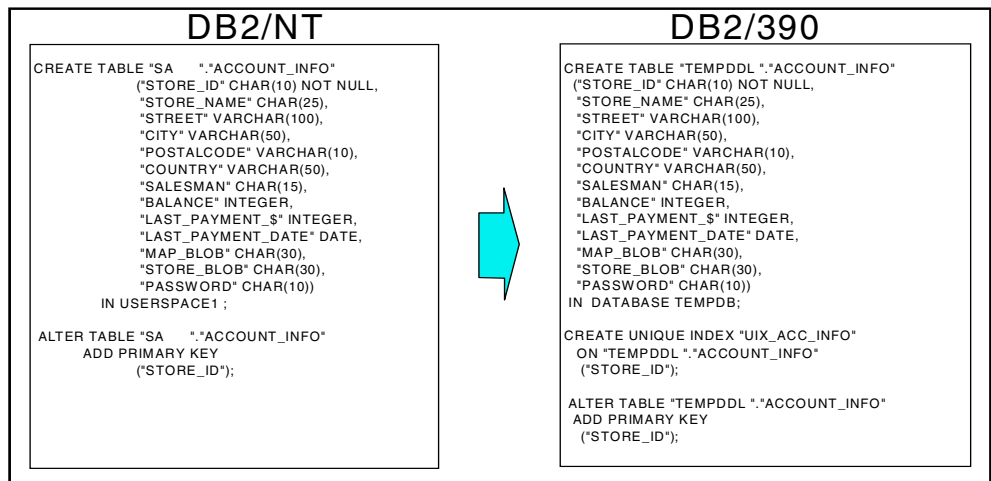


Figure 24. Translate table and index DDL

4.1.5.4 Translate the view DDL

In our specific porting exercise, we have no need to port views from DB2 for Windows NT to DB2 for OS/390. This is fortunate, since porting views can be difficult. Within the SQL language set, there is a common level of syntax base that is implemented in most, if not all, RDB products. On top of this common base, however, there are often additional SQL constructs that have been added to a particular RDB product that may not exist in other RDB products. By using any such non-standard SQL constructs, you may need to do significant rework later in attempting to port your SQL to a different RDB platform. To maintain compatibility across different RDB products, adherence to the ANSI/ISO SQL entry level standard of 1992 is highly recommended.

4.1.6 Methods to migrate data

You may find, as we did, that the enabling of DRDA connections between the source and target RDB products is not only useful, but necessary. All of the methods we use in migrating data from Windows NT to OS/390 require DRDA connections. Using DB2 for Windows NT export/import, for example, requires that on import that the DB2 for OS/390 system behave as a database server to the DB2 for Windows NT system. Conversely, using the DSNIAUL method to migrate data requires that the DB2 for Windows NT RDB act as a server to a DB2 for OS/390 requestor.

4.1.6.1 DB2 for Windows NT export/import

You can use the export utility on DB2 for Windows NT to export data from a database to one of several external file formats. The details of the formats and the syntax are available online via the *information center* on DB2 for Windows NT. The export utility produces a warning message whenever a character column with a length greater than 254 is selected for export to DEL format files.

The import utility can be used to insert data from an external file with a supported file format into a table, hierarchy, or view. Integrated exchange format, PC version (PC/IXF) import should be used to move data between databases. If character data containing row separators is exported to a delimited ASCII (DEL) file and

processed by a text transfer program, fields containing the row separators will shrink or expand.

DB2 Connect can be used to export tables from DRDA servers, such as DB2 for OS/390, DB2 for VM/ESA and VSE/ESA, and DB2 for OS/400. Only PC/IXF export is supported. We use DB2 Connect to import data from PC/IXF files on Windows NT to tables on OS/390.

From the DB2 Command Line Processor (CLP) on Windows NT, the following command is issued to make the connection to a DB2 subsystem (db2p) on OS/390:

```
connect to db2p in share mode user <userid> using <password>
```

Insert your user ID and password to replace <userid> and <password>.

Once the connection is successful, the import utility can be run from the CLP with the help of the following command:

```
import from d:\ixf\shopper.ixf of ixf insert into miguser.shopper
```

Here, we are importing data into the SHOPPER table and the database owner name is miguser.

Note that if you decide to use import/export as the method of migration, you should refer to Appendix F, “Alternative approach to database migration” on page 107 for more information. Of particular interest is handling the order of table imports to respect referential constraints.

4.1.6.2 DB2 for OS/390 DSNTIAUL Sample and Load Utility

While DB2 for Windows NT export/import uses DB2 for Windows NT to push the data into DB2 for OS/390, there is also a reciprocal path whereby DB2 for OS/390 can pull the needed data in from DB2 for Windows NT. While there can be some variants of this process, we describe using the DB2 for OS/390 DSNTIAUL sample application to unload data and the DB2 for OS/390 Load utility to populate tables. There are a few good justifications for directing the migration process from the DB2 for OS/390 side:

- **Performance**

The DB2 for Windows NT Import method relies on SQL insert processing to move data into the targeted table. For larger tables, this can be an extremely inefficient method. The DB2 for OS/390 Load utility is the most efficient way to populate tables where the number of rows is significant. This is primarily because the DB2 Load utility has a far more efficient interface with DB2 than SQL insert and because logging can be disabled when the Load utility is used.

- **Referential integrity checking**

Using DB2 for Windows NT Import may be tricky when referential constraints exist on the tables being populated. This is because you must ensure that a parent record exists before attempting to insert a child record that is dependent on that parent. Determining the order of table population so as to avoid referential constraint problems can be difficult when you have many relations linking multiple tables. The DB2 for OS/390 Load utility, however, can optionally ignore referential constraint checking during the loading process. After loading the entire set of tables, referential constraint checking can then be performed all at once.

The DB2 for OS/390 Load utility requires a rather specific format for its input data set. Unfortunately, none of the DB2 for Windows NT export formats are usable as input to the DB2 for OS/390 Load utility without significant post-processing. DB2 for OS/390, however, supplies a sample unload program called DSNTIAUL. DSNTIAUL is an assembler language sample program that unloads tables into sequential data sets that can later be reloaded into the same DB2 table or into another DB2 table using the DB2 for OS/390 Load utility. As well as unloading data, DSNTIAUL also creates the Load utility syntax required to invoke the DB2 for OS/390 Load utility with the unloaded data sets.

Upon investigation, the DSNTIAUL application can be used to unload tables on remote DB2 systems. It was not designed to work in cases where the DB2 server is anything other than DB2 for OS/390, however. To allow for DB2 for Windows NT to be the server to DB2 for OS/390, we find that the DSNTIAUL SQL parameter can not be used to invoke the sample. This is because the parsing code of DSNTIAUL is compatible only with DB2 for OS/390 as the target database. Without the SQL parameter, DSNTIAUL requires that the input data set be a list of table names and optional predicates.

Using DSNTIAUL to perform DRDA unloads in an heterogeneous operating environment requires that you must use DB2 for OS/390 V5, which includes a bug fix that was provided by APAR PQ32311, or that you use DB2 for OS/390 V6 or higher where the problem has already been fixed.

There is another implication in not being able to use the SQL parameter of DSNTIAUL. This presents a slight problem in being able to connect to the DB2 for Windows NT remote server since the CONNECT TO <db_server> SQL statement can not be issued. To enable the DB2 for OS/390 application to use DB2 for Windows NT as a server, you must bind the DSNTIAUL plan and package to the DB2 for Windows NT database. The following is an example of such a remote bind:

```

BIND PACKAGE (FRIENDLY.REMOTE) MEMBER (DSNTIAUL) -
ACT(REP) ISO(CS) SQLERROR(NOPACKAGE) VALIDATE(BIND)

BIND PLAN (DSNTIAUL) PKLIST (FRIENDLY.REMOTE.DSNTIAUL) -
ACT(REP) ISO(CS) CURRENTSERVER (DEMOMALL)

```

After performing the remote bind, you must next prepare the input and output data sets for the DSNTIAUL invocation(s). To simplify the process, you should separate your input list of table names into two separate lists: One for new tables that are to be added to the DB2 for OS/390 environment, and the second for those base Net.Commerce DEMOMALL tables that already exist on both platforms. To generate the list of new user tables that will be added, you can slightly modify the query in 4.1.2.1, "Query to locate new tables and views" on page 58 to extract only the creator and table name. To generate the list of those Net.Commerce tables where data will be moved, you can make the same modification to the query used in 4.1.3, "Common tables of Windows NT and OS/390 Net.Commerce" on page 62.

For each invocation of DSNTIAUL, you can unload up to 100 tables. Each table will be unloaded to its own unique data set, defined by the JCL DD statements SYSREC00, SYSREC01 to SYSREC99, where the numeric portion of SYSREC is determined by the position of the table being unloaded in the SYSIN input list. The Load utility syntax for all tables unloaded will be placed into the data set defined by the SYSPUNCH DD statement. An example of the JCL that can be

used to execute the DSNTIAUL unload follows, where four tables are being unloaded (SYSREC00-SYSREC03) and the list of tables being unloaded is contained in a separate file named HAUSER.SPUFI.IN(TABLIST1):

```
//UNLD1 JOB CLASS=A,MSGCLASS=X,REGION=6M,NOTIFY=&SYSUID
//UNLD EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPUNCH DD DSN=FF.UNLD1,UNIT=SYSDA,SPACE=(CYL,(1,10),RLSE),
// DISP=(NEW,CATLG)
//SYSREC00 DD DSN=FF.UNLD00,SPACE=(CYL,(5,10),RLSE),DISP=(NEW,CATLG)
//SYSREC01 DD DSN=FF.UNLD01,SPACE=(CYL,(5,10),RLSE),DISP=(NEW,CATLG)
//SYSREC02 DD DSN=FF.UNLD02,SPACE=(CYL,(5,10),RLSE),DISP=(NEW,CATLG)
//SYSREC03 DD DSN=FF.UNLD03,SPACE=(CYL,(5,10),RLSE),DISP=(NEW,CATLG)
//SYSTSIN DD *
DSN SYSTEM(DB2P)
RUN PROGRAM(DSNTIAUL) PLAN(DSNTIAUD) -
LIB('DB2V510P.RUNLIB.LOAD')
END
//SYSIN DD DSN=HAUSER.SPUFI.IN(TABLIST1),DISP=SHR
//
```

At this point, you have not modified the contents of the existing OS/390 DEMOMALL base. You should perform full image copies of all the tablespaces in the database at this point. This provides a point of consistency to which you can restore back easily if you encounter problems during the loading of the ported data into the tables. The DB2 Copy utility, with the FULL YES parameter, should be used to backup the newly created DEMOMALL.

After executing the DSNTIAUL unloads, you will need to edit the resulting Load utility syntax. There are several Load utility options that you may need to add or change in the DSNTIAUL generated syntax for each individual Load invocation:

- Edit the table names referenced in the Load INTO TABLE parameter, if needed, so that the creator and table name refers to the OS/390 table object you wish to port the data into.
- The parameter ENFORCE NO should be added. This parameter specifies that no referential integrity checking should be undertaken during the load process. By using this parameter, you can load sets of tables without needing to order them such that parents are loaded before children. By using this parameter, the tablespace will be placed in Check Pending status. Later, you will need to use the Check utility to perform referential integrity checking and to turn off the Check Pending status.
- Provide the ERRDDN, MAPDDN, and DISCARDN parameters and data sets. These become important in a situation where you will be loading data from the common Net.Commerce for Windows NT tables into their OS/390 siblings. Unique indexes will cause discarding of duplicate entries into these added output files. This is beneficial in the cases, such as Net.Commerce-provided commands, where the OS/390 versions are the correct ones. However, for cases where the user may have added commands that inadvertently duplicate one provided in OS/390, the duplicate will also be put into the discard data set. To discern between those entries that are rightly discarded and those that will need to be changed and inserted into the OS/390 table, you will need to perform analysis.
- Use the REPLACE and RESUME parameters appropriately. For new user tables, RESUME NO and REPLACE YES will allow you to run the Load utility (multiple times if need be for debugging) without accumulating rows due to reloading. For existing Net.Commerce tables, you will need to use RESUME

YES and REPLACE NO to ensure that you do not eliminate the existing rows in the tables.

- The final requirement is to provide the required JOB, EXEC, and DD statements. The following sample job can be used as a general guideline, but you may need to change the data set names and DD names to match those required by your own scenario. Space allocations for the data sets may also need to be changed to fit the needs determined by the size of your particular situation.

```
//LOAD1 JOB CLASS=A,MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//*
//UTIL EXEC DSNUPROC,SYSTEM=DB2P,UID='HAUSER.LOAD1'
//*
//SORTWK01 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTWK02 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTWK03 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTWK04 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTOUT DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SYSERR DD DSN=MIGR.SYSERR,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(2000,(20,20),,,ROUND)
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSMAP DD DSN=MIGR.SYSMAP,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(2000,(20,20),,,ROUND),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSDISC DD DSN=MIGR.SYSDISC,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(2000,(20,20),,,ROUND),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
//SYSREC00 DD DSN=MIGR.UNLD00,DISP=SHR
//DSNUPROC.SYSIN DD *
LOAD DATA LOG NO INDDN SYSREC00 RESUME YES ENFORCE NO
ERRDDN SYSERR MAPDDN SYSMAP DISCARDN SYSDISC
INTO TABLE TEMPDDL.ACCOUNT_INFO
(
STORE_ID POSITION( 1 )
CHAR( 10) ,
STORE_NAME POSITION( 11 )
CHAR( 25)
NULLIF( 36)='?',
STREET POSITION( 37 )
VARIABLE
NULLIF( 139)='?',
CITY POSITION( 140 )
VARIABLE
NULLIF( 192)='?',
POSTALCODE POSITION( 193 )
VARIABLE
NULLIF( 205)='?',
COUNTRY POSITION( 206 )
VARIABLE
NULLIF( 258)='?',
SALESMAN POSITION( 259 )
CHAR( 15)
NULLIF( 274)='?',
BALANCE POSITION( 275 )
INTEGER
NULLIF( 279)='?',
"LAST_PAYMENT_ $" POSITION( 280 )
INTEGER
NULLIF( 284)='?',
LAST_PAYMENT_DATE POSITION( 285 )
DATE EXTERNAL( 10)
NULLIF( 295)='?',
MAP_BLOB POSITION( 296 )
CHAR( 30)
NULLIF( 326)='?',
STORE_BLOB POSITION( 327 )
CHAR( 30)
NULLIF( 357)='?',
PASSWORD POSITION( 358 )
CHAR( 10)
NULLIF( 368)='?'
)
```

After executing the Load utility jobs to migrate your data, you will need to check the output to verify that all has gone normally. If rows have been discarded during any of the loads, you will need to verify that these discarded rows were provided by Net.Commerce for Windows NT. Any discarded rows that are not Net.Commerce for Windows NT-provided will need to be analyzed, changed, and inserted, since they represent user created data. Return codes for the Load utilities should be no larger than four, due to the fact that tablespaces will be left in Copy and Check Pending states.

- **LOADTBLS**

The loadtbls utility is described in Appendix B of the *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862. It is invoked with the MVS_BUILDDDB.SH shell script that is present in the /usr/lpp/NetCommerce/bin directory. We use the command in the following manner:

```
mvs_buildddb.sh -c -i ncommerce.conf -tc -v table_name
```

Where ncommerce.conf is the file created by the cmnconf utility when creating an instance and is stored in the /usr/lpp/NetCommerce/html/en_US/<instance_name> directory.

The ncommerce.conf file must have an entry for MS_DELPATH that gives the fully qualified name of the directory that contains the custom delimited (del)¹ files from a non-OS/390 system. For example:

```
MS_DELPATH /usr/lpp/NetCommerce/instance/ffinst/del
```

The MVS_BUILDDDB.SH converts the del file to a delimited format suitable for OS/390 called mvssel and then invokes the loadtbls utility that reads the mvssel file and inserts each record into table table_name as defined above.

4.1.7 Database housekeeping

Regardless of the method you use to transport your data from Net.Commerce for Windows NT to Net.Commerce for OS/390, there are DB2 for OS/390 procedures you should perform to verify, organize, and back up the migrated objects as well as to prepare for their optimal use.

4.1.7.1 The Check Data utility

The Check Data utility verifies that the contents of a set of tablespaces abide by the referential constraints that are defined in the tables they contain. If, after migrating to OS/390, any of the tablespaces are left in Check Pending status, you must run the Check Data utility. Since referential constraints are defined in the Net.Commerce tables, it seems reasonable that even if tablespaces are not overtly in Check Pending status that executing the Check Data utility for all tablespaces provides a checkpoint for data assurance.

For our sample migration, a single Check Data utility job was executed. You can perform the checks using multiple utility executions, where each separate job represents one or more tablespace sets. To create the utility syntax, the tablespace list was generated by a simple SQL Select query of all tablespaces in the Net.Commerce database:

```
//CHECK JOB CLASS=A,MSGCLASS=X,REGION=0M,NOTIFY=&SYSUID
//*
//UTIL EXEC DSNUPROC,SYSTEM=DB2P,UID='HAUSER.CHECK'
//*
//SORTWK01 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTWK02 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
```

¹ When exporting from a non-OS/390 system, use a date format of DATESISO. This ensures compatibility with DB2 for OS/390.

```

//SORTWK03 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTWK04 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SORTOUT DD UNIT=SYSALLDA,SPACE=(CYL,(50,50))
//SYSERR DD DSN=MIGR.CHECK.ERRORS,DISP=(NEW,CATLG,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(5,1),RLSE)
//DSNUPROC.SYSIN DD *
CHECK DATA
    TABLESPACE TEMPDB.ACCOUNTR
    TABLESPACE TEMPDB.ACCRCMDG
    TABLESPACE TEMPDB.ACCRGROU
    TABLESPACE TEMPDB.ACCRMODE
    .....
    TABLESPACE TEMPDB.USRTTRAFF
    TABLESPACE TEMPDB.WTAXMERI
    SCOPE ALL

```

4.1.7.2 Reorg/Copy or Copy utilities

After verifying the contents of the migrated database using the Check Data utility, you should back up the tablespaces using the Copy utility. You can also, optionally, reorganize the data in the tablespaces and indexes to provide an optimal starting point for performance and space utilization by using the Reorg utility. For our sample migration, we opted to perform both a reorganization and a copy of the Net.Commerce system. The Reorg and Load utilities both contain the option of generating image copy data sets in parallel to the Reorg or Load task. In this example, by using the COPYDDN option combined with the REORG utility, the elapsed time to perform the image copy and the reorganization is only slight more than that of the reorganization task alone.

For our sample migration, the Reorg utility job and syntax was generated with REXX (see Appendix E.1, “Reorganize and copy REXX sample” on page 105) using the tablespace list as used for the Check Data utility execution:

```

//REORGTS JOB CLASS=A,MSGCLASS=X,REGION=5M,NOTIFY=&SYSUID
//R001 EXEC DSNUPROC,SYSTEM=DB2P,UID='REO001'
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(15,15))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(15,15))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(15,15))
//SORTWK04 DD UNIT=SYSDA,SPACE=(CYL,(15,15))
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(15,15))
//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(15,15))
//SYSPRINT DD SYSOUT=*
//SYSCOPY DD DSN=MIGR.COPY001,DISP=(NEW,CATLG,DELETE),
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DCB=(RECFM=FB,LRECL=4096,BLKSIZE=24576,BUFNO=30)
//SYSREC DD DSN=MIGR.RECS001,DISP=(NEW,DELETE,KEEP),
// UNIT=SYSDA,SPACE=(CYL,(5,5),RLSE),
// DCB=(RECFM=FB,LRECL=4096,BLKSIZE=24576,BUFNO=30)
//SYSIN DD *
REORG TABLESPACE TEMPDB.ACCOUNTR LOG NO
SORTDATA SORTKEYS COPYDDN(SYSCOPY)

```

4.1.7.3 Runstats

The final recommended maintenance step for the newly migrated Net.Commerce database is to collect statistics on all the tablespace and index objects. In DB2 for OS/390, the database statistics are collected using the Runstats utility. Statistics are collected and placed into the DB2 system catalog. These statistics are used during the query bind process to help determine the optimal access path for query execution. Therefore, after collecting statistics, you should always rebind static plans to assure that suboptimal access paths are not retained.

There are numerous invocation options for the Runstats utility. For our sample migration, we ran the Runstats utility set to collect all statistics on the tablespaces, tables, and indexes as shown in the following sample. We generated the Runstats utility JCL and syntax, again, using REXX (see Appendix

E.2, “RUNSTATS REXX sample” on page 105) and an input list of all the tablespaces in the migration.

```
//STATS JOB CLASS=A,MSGCLASS=X,REGION=5M,NOTIFY=&SYSUID
//E001 EXEC DSNUPROC,SYSTEM=DB2P,UID='STAT001'
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
RUNSTATS TABLESPACE TEMPDB.ACC//SYSMAP DD
DSN=SAMPJOB.LOAD.STEP1.SYSMAP,DISP=(MOD,DELETE,CATLG),
// UNIT=SYSDA,SPACE=(2000,(20,20),,ROUND),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2400)
OUNTR TABLE INDEX
```

4.2 Summary

In this chapter, we discussed the stages in migrating a copy of the Windows NT production Net.Commerce database to Net.Commerce for OS/390. The stages involved are as follows:

1. Identify the differences between a copy of the production Windows NT Net.Commerce database and the OS/390 DEMOMALL database.
2. Use DB2LOOK on Windows NT to create DDL for the Windows NT database tables.
3. Execute this DDL from Windows NT to create and update the necessary tables in the OS/390 DB2.
4. From OS/390, use the DSNTIAUL DB2 sample to connect to the remote Windows NT server (via DRDA) and unload the data in the remote DB2 database to an OS/390 data set.
5. From OS/390, use the DB2 Load utility to load the data from the data set into DB2 itself.
6. Run DB2 housekeeping.
7. Test the migrated/porting solution (function and performance).

Chapter 5. Application development

This chapter describes the platform differences from the perspective of an application developer. We document the major differences in the tools available for developing Net.Commerce C++ Commands and Overrideable Functions (OF) between the distributed and OS/390 environments.

Development of HTML pages and Net.Data macros are not covered here; they are assumed to be generated with distributed workstation tools (such as the WebSphere Visual Studio and the Net.Commerce Template Editor) and then transferred to the OS/390 server.

Net.Commerce is customized in many cases since there is not a suitable Command or OF within the base product to meet the customers specific business requirement. In these cases, a completely new Command or OF has to be created. With version 3.1.2, Net.Commerce Commands and Overrideable Functions are written in C++.

Application development tools are constantly changing, but generally, developers of OS/390 Net.Commerce solutions appear to be comfortable using distributed workstation tools to produce the C++ source files. These source files are then transferred to the OS/390 environment (typically by ftp) and then compiled on OS/390.

Distributed workstation tools do not have to be used to create the C++ source files. OS/390 itself can be used to create these files using the OS/390 ISPF editor or the OS/390 UNIX System Services *vi* editor.

The OS/390 development environment is very similar to a traditional UNIX command line environment. If the application developers involved in the port of a Net.Commerce application from Windows NT or UNIX to OS/390 are familiar with the UNIX command line environment, they should have few migration considerations when moving to OS/390. If the application developers only have experience with Windows GUI application development tools, the changes between the development environments will be more noticeable.

Remember that Net.Commerce Commands and Overrideable Functions compile to dynamic link libraries, or *.dll files on Windows NT and to shared objects or *.so files on UNIX and OS/390 systems.

5.1 Microsoft 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. An example of a makefile to build a Net.Commerce Command on Windows NT is shown in Appendix D, "OS/390 Net.Commerce development tools" on page 97.

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 run-time environment for DLLs, where resources the DLL needs are not available until it is loaded and running in the Net.Commerce environment. Figure 25 shows a sample screen of the Windows NT Microsoft Visual C++ 6.0 environment.

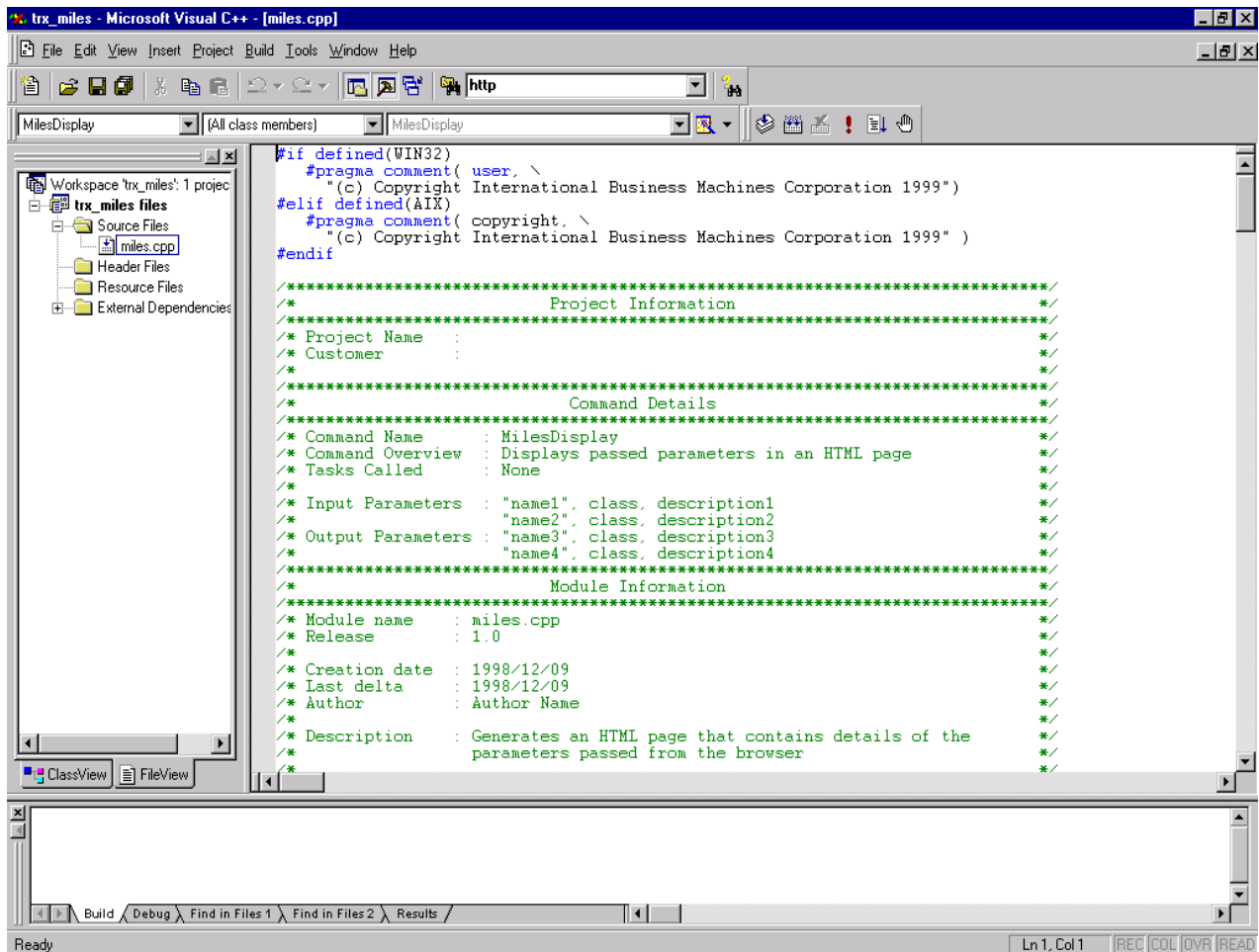


Figure 25. Microsoft Visual C++ 6.0 compilation of a Net.Commerce Command

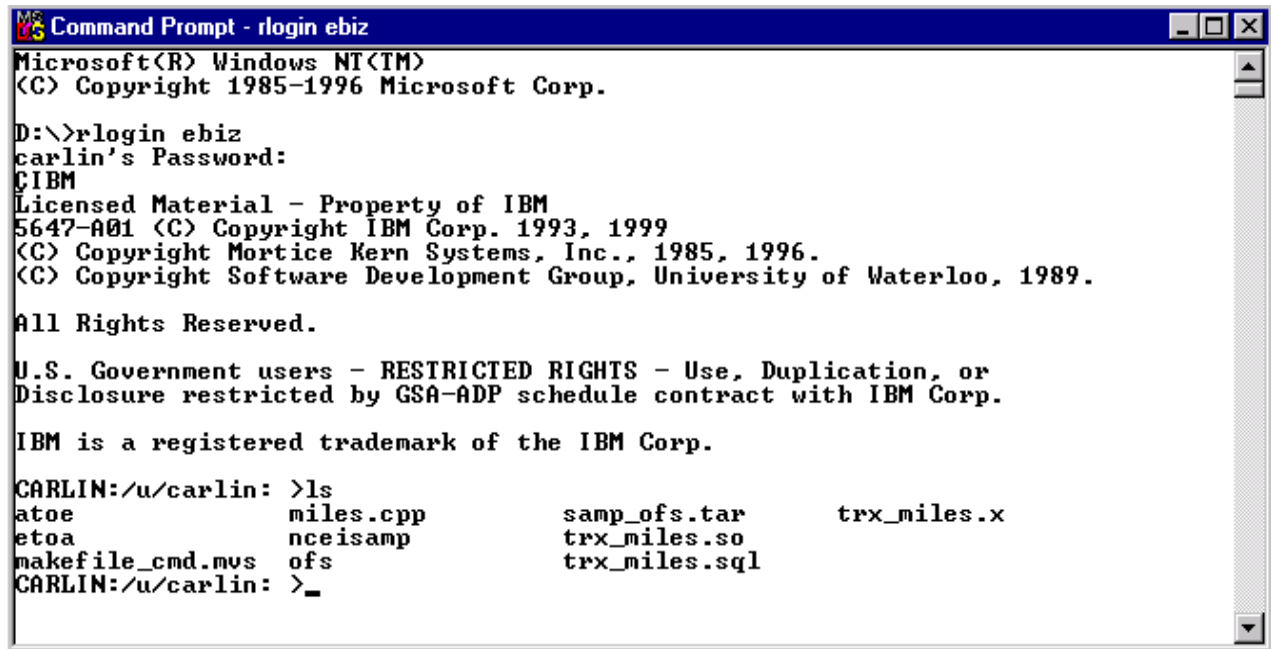
This is a GUI interface with everything that the environment offers, such as graphical editors with drag and drop capabilities.

5.2 UNIX and OS/390 command line development

UNIX and OS/390 UNIX System Services have a shell-like interface, similar to the Command Window on Windows NT.

On OS/390, there are at least three ways to start a UNIX System Services shell. The first, and probably the most common for a C++ developer, is to use rlogin. The second is to log on to TSO and then run the OMVS (perhaps from TSO option 6) command. The third is to use telnet to access the OS/390 server.

An example of an rlogin to the OS/390 UNIX System Services shell on OS/390 is shown in Figure 26. The rlogin session is initiated from Windows NT workstation (using a freeware rlogin tool) to an OS/390 server (called ebiz).



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

D:\>rlogin ebiz
carlin's Password:
CIBM
Licensed Material - Property of IBM
5647-A01 (C) Copyright IBM Corp. 1993, 1999
(C) Copyright Mortice Kern Systems, Inc., 1985, 1996.
(C) Copyright Software Development Group, University of Waterloo, 1989.

All Rights Reserved.

U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or
Disclosure restricted by GSA-ADP schedule contract with IBM Corp.

IBM is a registered trademark of the IBM Corp.

CARLIN:/u/carlin: >ls
atoe          miles.cpp      samp_ofs.tar   trx_miles.x
etoe          nceisamp      trx_miles.so
makefile_cmd.mvs ofs           trx_miles.sql
CARLIN:/u/carlin: >
```

Figure 26. rlogin session to OS/390

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

Files in the OS/390 UNIX System Services environment reside in an hierarchical file system (HFS). The Net.Commerce C++ source files are typically HFS files.

There are many tools to edit HFS files. UNIX programmers will be familiar with the vi editor. To use vi on OS/390, you first have to export a valid TERM type in your rlogin session, because 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
```

OS/390 programmers may be more familiar with the TSO and ISPF interface. ISPF edit sessions of HFS files can be initiated in a number of ways.

A TSO session with an OS/390 system can be started in many different ways; products, such as e-Network Host on Demand and e-Network Personal Communications from IBM, can be used to initiate a session through the tn3270 or SNA protocols.

Once logged onto TSO, a common way to edit an HFS file is to type `ishell` from TSO Option 6. The ISPF editor is a full screen editor. For those unfamiliar with the OS/390 ISPF editor, we provide a sample screen shot in Figure 27 on page 80.

```

HSTM3 - A - TCP00001
File Edit Transfer Appearance Communication Assist Print Help

BROWSE -- /u/carlinn/makefile ----- Line 00000000 Col 001 071
Command ==> ----- Scroll ==> CSR
***** Top of Data *****
# -----
# Some Definitions
# -----

CC      = cxx
NC_ROOT = /usr/lpp/NetCommerce
DB2PATH = /usr/lpp/db2/db2510
LIBPATH = $(NC_ROOT)/lib/

LE_HLQ  = PP.ADL370.OS390R7D

MYSINC  = -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 \
          F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
          F7=UP        F8=DOWN       F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE

MA a
02/015

```

Figure 27. ISPF editor session

5.3 OS/390 C++ compilation tools

On OS/390, you use the `cxx` command to compile C++ source code. More typically, you will 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.

5.3.1 Example of an OS/390 makefile

Creating a valid makefile for OS/390 can be time consuming, so we have provided one for Net.Commerce Commands and Overrideable Functions in Appendix D.1, "OS/390 Net.Commerce sample makefile" on page 97.

The key points to note in this makefile is that compiler option `LANGVLV(EXTENDED)` must be used and `DLL` is specified as a link option. Note also that this makefile has all the 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.

5.4 DB2 application development on OS/390

When you write Net.Commerce Commands and Overrideable Functions they use the Net.Commerce C++ SQL class to access the Net.Commerce DB2 database.

A typical use of the SQL class to execute an update to a DB2 table might look like this:

```
String.sqlStmt; Row.row;  
sqlStmt.= "update MYTABLE set MYVAR='A' where MYORNBR=";  
sqlStmt << order_m;  
SQL sql01( *( DataBaseManager::GetCurrentDataBase()), sqlStmt);
```

5.4.1 SQL from C with Net.Commerce V3.1 and V3.1.1

Before Version 3.1.2 of Net.Commerce on OS/390, the C++ Command structure and database class was not available. If you are in a V3.1 or V3.1.1 Net.Commerce environment and are building integration code that uses static SQL to access DB2, you will need to first precompile this code with the C DB2 pre-compiler. You do not have to do these steps if you are in a V3.1.2 environment.

On System/390, the precompile step removes the static SQL statements from the C code and creates a database request module, or *DBRM*, which is usually called a *bind file* on distributed DB2 platforms.

An example of a REXX exec that runs the C DB2 precompiler is in Appendix D.2, "Precompiling C code containing static SQL with Net.Commerce V3.1.1" on page 99.

On System/390 the DBRMs are attached to *PLANS* through a *BIND* process. A PLAN is an executable DB2 object that can contain DBRMs and DB2 *PACKAGES* (which in themselves can contain DBRMs).

A DB2 bind job should be run to bind the DBRM created in the precompile step into the Net.Commerce DB2 plan that you are using. A sample bind JCL job is available in the Net.Commerce samples data set; in fact, this bind job has to be run to install Net.Commerce. For further information about running the sample bind job, see *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862.

If your program is using dynamic rather than static SQL, you can code and run your programs without precompilation if you use the DB2 call level interface (CLI). DB2 on OS/390 has a prebound plan for programs wishing to use dynamic SQL via the CLI; the name of the plan is DSNAOCLI.

However, static SQL performs better since expensive run-time statement PREPAREs are not needed.

Appendix A. Sample customer scenario for migration and porting

The approach taken is to first copy the Net.Commerce solution from the production Windows NT server to a test Windows NT server and then to migrate this test copy to OS/390. The benefits of this approach are that the production Windows NT Net.Commerce site can still do business and a copy of this site can be ported to OS/390 in a controlled and tested manner.

A.1 Copying a Windows NT Net.Commerce database

The source Windows NT Net.Commerce application we copied was built on the sample DEMOMALL instance provided with Net.Commerce Version 3.1.1 on Windows NT.

The transfer of the database between the two Windows NT systems was accomplished in the following manner:

On the source Windows NT:

1. The html, image, and macro files were zipped along with their relative path information into one zip file.
2. The database was backed up into a single file using the DB2 Control Center. The backup utility will create this single image file in a directory structure starting with \<database name>.0\.
3. This single file was sent to the target Windows NT machine by ftp.

On the target Windows NT:

1. Place the single image of the database into the same directory structure starting \<database name>.0\.
2. The database was restored using the following command from the DB2 Command Line Processor:

```
db2> restore database <database name> from c:\ taken at 19990916150027 to c:  
into demomall with 2 buffers buffer 1024
```

This assumes that the backup image is in c:\<database name>.0\.
3. The zip file containing the html, image, and macro files was unzipped.
4. Working with the new copied database, you may see an authorization problem when trying to access the database. To avoid this, make sure you are logged on to a Windows NT user ID that has DB2 administration privileges and is the same name as the DB2 administration user ID from the Windows NT system that contains the source database. For example, on the source Windows NT the Net.Commerce database was created with DB2 administration user ID fred. Make sure that on the target Windows NT system you are logged on as fred and that fred has DB2 administration privileges.
5. A new instance was created on the target Net.Commerce system with the following parameters:
 - html path: c:\ibm\www\html\en_US\<database name>
 - macro path: c:\ibm\NetCommerce3\macro\en_US
 - database owner: sa
 - database password: password
6. The hosts file in C:\WINNT\system32\drivers\etc was updated to include the hostname for the new instance.
7. The image files from c:\ibm\NetCommerce3\html\en_US\<instance name> were copied to c:\ibm\www\html\<database name>\<instance name>.

Appendix B. Solutions Assurance and stress testing

Solutions Assurance and stress testing are important aspects of any OS/390 Net.Commerce implementation. If carried out correctly, these activities will greatly improve the quality of the migrated solution and reduce costs due to unscheduled outages once the site is in production.

B.1 Solutions Assurance

In today's world of ever-evolving technological advances, it is easy to believe that poorly implemented e-business solutions are due to technology problems. However, the reality is that most mission-critical problems evolve from the lack of proper analysis, planing, project management, and having all the correct people involved.

Included in a Solutions Roadmap are techniques in understanding a customer's e-commerce strategy and goals, skills required to implement the solution, services options available for installation of various components, and most importantly, how to acquire the appropriate skills to ensure success.

B.2 Stress testing

Net.Commerce stress/load testing is necessary before introducing your e-commerce site to the world. The traffic generated through the Internet, especially for those that are commerce related, can have various effects on your system. Before making your site public, your site should be heavily tested in relation to the traffic that is expected once the site goes live.

B.2.1 Stress test

A stress test focuses on a break or degradation of performance at a point in time based on a constant input. For example: How does a constant number of concurrent users accessing the site cause the system to react over time? See Figure 28 on page 86.

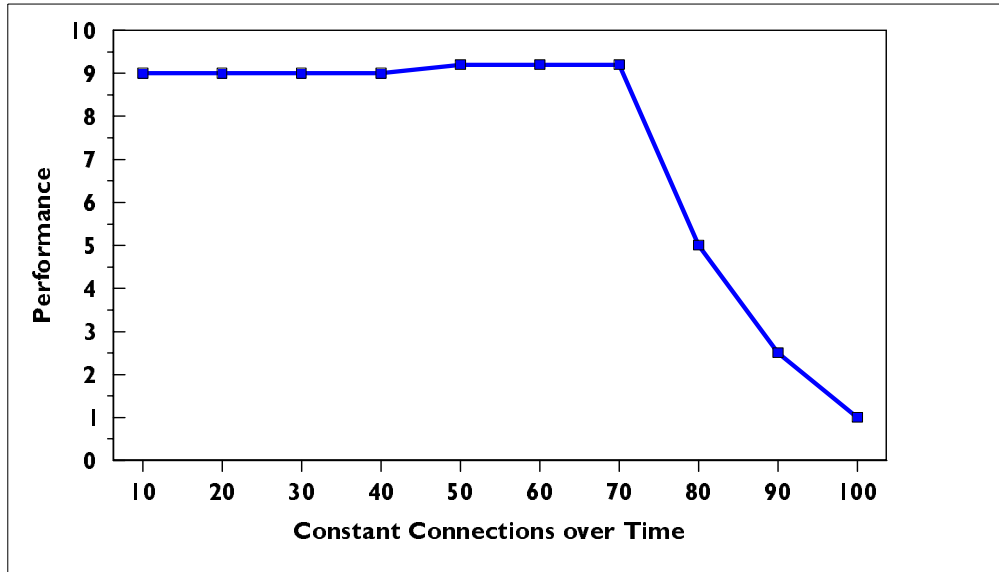


Figure 28. Generic stress test

B.2.2 Load test

A load test focuses the intention of measuring a break in functionality based on variable input with the amount of data remaining constant. For example: How does the number of concurrent users accessing the site cause the system to react? See Figure 29.

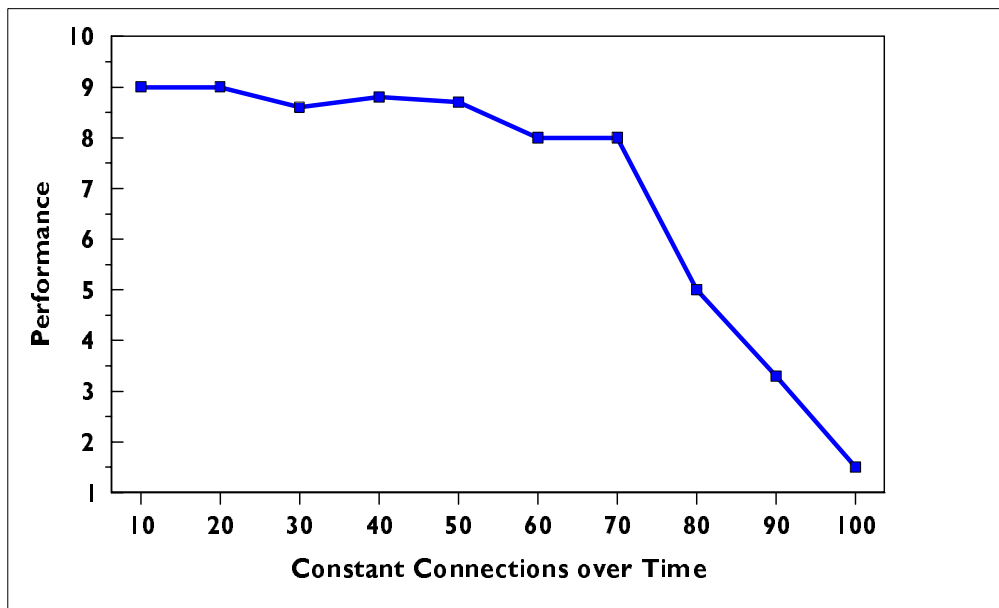


Figure 29. Generic load test

B.3 Pretest strategy

The first step in performing a Net.Commerce stress/load test is to determine what “typical” user traffic is for your site. Will this be a business-to-business site where

customers will be predominantly browsing? Will this be an Internet site open to the general public for browsing, shopping, and buying? This information will be used when you further begin your test script development.

Next, you need to make certain assumptions based on the expected volume of hits to your Net.Commerce site, as well as some goals you hope to achieve in your testing. The expected volume would include the number of registered users, expected transactions per day/week/month, and a breakdown of the types of transactions. Goals could be measured again in the number of transactions per day/week/month and response time. Once these assumptions are made, you are now ready to start developing your test scripts.

B.4 Script development

The basic building blocks for load/stress testing are scripts. Scripts are unique to a customer's Web site and expected traffic, but general guidelines are applicable to most commerce-related sites. In a generalized Net.Commerce setup, there are three types of scripts: BROWSING, SHOPPING, and BUYING. BROWSING scripts represent those that do not use SSL or Authentication, SHOPPING scripts represent those that use SSL but no authentication, and BUYING scripts use both SSL and authentication. Each simulated end-user shopping scenario is a script. Scripts are combined to create testcases. Testcases allow for a more realistic simulation of Web traffic. Each testcase should consist of various ratios and combinations of scripts. For instance, a testcase might consist of 80 percent BROWSING, 10 percent SHOPPING, and 10 percent BUYING. If a testcase is executed with 10 concurrent users, eight will be browsing, one will be shopping, and one will be buying. The industry averages a 10 percent buying rate; however, you need to decide if this is an acceptable assumption in your Net.Commerce shop. This is where you can incorporate the information derived from your pretest strategy. The heaviest stress on a system would be that which simulates 100 percent buying. This would not only stress the Web server, but Net.Commerce and your DB2 subsystem as well.

B.5 Outline of test runs

Once your scripts and testcases have been defined, it is good practice to outline your test runs. First, you should test with one user performing each script to test functionality. The next step is to test with two concurrent users. Net.Commerce generates unique order reference numbers, and depending on how you develop your store, you might need to parse out other values during the script to pass out later. Thus, the two user run will catch any other problems with the functionality of the script in acquiring unique IDs or logging in as unique users. Once your scripts are tested to verify concurrency, you can start running your testcases incrementally increasing users to maximum CPU as well as running with increased elapsed time.

For a load test where virtual users will be incremented exponentially until the system has reached its maximum, performance data needs to be recorded at each increase of users to predict how heavier loads will affect the system. With a stress test, data also needs to be recorded after periods of continuous, steady stress in order to predict how the system will react to heavy, continuous traffic

(such as overnight test runs). These two methods of testing will allow analysis of both load (incremented users) and stress (incremented time).

B.6 Stress test environment

For a good stress test, an intranet token-ring LAN is not adequate. In a stress test, you want to stress the system, not the connection. You do not want any other users on that network to corrupt the performance results and you want to prevent a bottleneck in the network connection. Token ring has a bandwidth of 16 Mb/sec. whereas Fast Ethernet has a bandwidth of 100 Mb/sec. For a stress test, they would want to use a Private Fast Ethernet or greater connection from their stress test (injector) system to their S/390 platform. This will create an isolated test bed that will lower the chance of a network bottleneck.

In choosing a user simulator tool, you need the ability to handle various functions. The tool you choose needs to be able to support a unique user per thread, which is necessary for a good representative script. You also need to be able to handle redirection, as well as the parsing of responses, which is necessary to get information regarding a customer to pass on through the script.

Appendix C. Net.Commerce platform function comparison

Table 3 compares functions available in Net.Commerce on OS/390, Windows NT (shown as “NT” in the column heading), and AIX.

Table 3. Net.Commerce engine functions

Function	V3.1.2		V3.2	Note
	OS/390	NT/AIX	NT/AIX	
Multiple store “Mall” structures	X	X	X	
Shopping carts, multiple shipto, and so forth	X	X	X	1
Commands, Tasks and Overrideable Functions (OFs)	X	X	X	2
Command and OF class libraries	X	X	X	
Create new Commands	X	X	X	
Create new Tasks	X	X	X	
Create new Overrideable Functions	X	X	X	
Customize shopping flow	X	X	X	
Net.Data	X	X	X	
Customize backend integration	X	X	X	3
Support MQSeries, EDI, CICS, IMS	X	X	X	
Backend server	X	X	X	
Dynamic page caching	X	X	X	
Multiple currency display support	X	X	X	
Euro currency support	X	X	X	
Translated versions	X	X	X	4
Automatic update of cached pages		X	X	5
Database schema provided	X	X	X	
Referential integrity	X	X	X	
Customizable database	X	X	X	
Multi-vendor database support		X	X	6
“Tier” structure	2	2 or 3	2 or 3	7
Multi-vendor HTTP server support		X	X	8
Web server/Net.Commerce interface	API	API	API	
Multiple server configuration	X	X	X	9
Multi-server load balancing	X	X	X	10
Automatic failover	X	X	X	
Use of cookies	X	X	X	

Function	V3.1.2		V3.2	Note
	OS/390	NT/AIX	NT/AIX	
Defer guest shopper creation until item placed in shopping cart			X	
User ID/password protection for shoppers	X	X	X	
SET payment for customer wallet	X	X	X	
Merchant-originated SET payment	X	X	X	
Integration with payment server	X	X	X	11
Configurable SSL	X	X	X	
Configurable authentication	X	X	X	
Password storage in database	X	X	X	
Password storage in RACF	X			
Logon/logoff capability	X	X	X	
ISV plugins available		X	X	12
Encrypted password storage	X	X	X	
Prevent user ID information from flowing on non-SSL link	X			

Table 4 compares the available utilities.

Table 4. Utilities

Function	V3.1.2		V3.2	Note
	OS/390	NT/AIX	NT/AIX	
Configuration utility	X	X	X	13
Store Administrator	X	X	X	
Site Administrator	X	X	X	
Template Designer	X	X	X	
Access control lists	X	X	X	
Product Advisor	X	X	X	
Catalog Architect		NT only	NT only	14
Store Creator	X	X	X	
Hosting Server		X	X	
DB2 Text Extender		X	X	
Mass import (massimpt)	X	X	X	
Database cleanup (ncclean)	X	X	X	
Performance monitor	X		X	
Multilevel tracing facility	X		X	

Function	V3.1.2		V3.2	Note
	OS/390	NT/AIX	NT/AIX	
Dynamic start/stop of tracing and performance monitoring	X			

Note	Detail
1	This level of functionality is identical between the platforms.
2	The Net.Commerce (NC ²) architecture is now common between the platforms. This is a key point, because it allows for a greater degree of portability between the platforms.
3	Slightly richer function in Windows NT/AIX.
4	Japanese only for OS/390.
5	OS/390 Net.Commerce must first implement support for DB2 UDB triggers. Since DB2 UDB for OS/390 was only very recently released, this support is not yet available.
6	OS/390 Net.Commerce relied on DB2 exclusively.
7	OS/390 Net.Commerce requires Web server, Net.Commerce, and DB2 to reside on the same machine. This is for performance reasons.
8	OS/390 Net.Commerce relies on WebSphere (IBM HTTP server).
9	OS/390 Net.Commerce can achieve this with multiple instances running on the same OS/390 system, or running in separate systems and connected via Parallel Sysplex.
10	OS/390 relies on the WorkLoad Manager (WLM).
11	Net.Commerce on Windows NT/AIX is very tightly coupled; on OS/390, it is more loosely coupled (providing greater flexibility to use latest version of Payment Server).
12	IBM is actively working with Independent Solution Vendors (ISVs) to port more of their applications to OS/390 to work with Net.Commerce on this platform.
13	Different instances, but same result.
14	Output from Windows NT version can be used with OS/390 Net.Commerce.

C.1 Differences between system architectures

Operating system architectures can be classified as one of the following models:

Monolithic Model	This model uses functions or procedures that can call any other procedure. It is not modularized. Maintenance becomes very difficult.
Layered Model	This model involves organizing the operating system into layers of code. This provides for easy maintenance and debugging.
Micro kernel Model	This model separates the operating system services, such as process management, virtual memory

management, device management, network management, and file system services, from the kernel. The kernel only provides basic services like interrupt, I/O, task and thread management, and interprocess communication. This makes the kernel easily portable to other hardware platforms.

Client/Server Model In this model, the operating system is divided into several processes. Each process (server) implements a single set of basic operating system services, such as memory services, network services, file services, and so forth. Clients can be another operating system component or an application program that requests services from the servers by sending messages.

They all have something in common. They all run tasks in two modes: the user mode (problem state in OS/390) and the kernel mode (supervisor state in OS/390). Tasks that run in kernel mode can access system hardware and system data. Applications usually run in user mode.

C.1.1 AIX features

The Advanced Interactive eXecutive (AIX) operating system is a layered system. It was a 32-bit system until Version 4.2. With Version 4.3, it is now a 64-bit system. The range of address ability in the 64-bit address space is one million terabytes (TB) - 2^{60} bytes. More than a third of this range has been reserved for application program text and data. Shared memory regions, shared libraries, dynamically loaded programs, and the program stack are *each* allocated a subset of the address space up to 2^{56} bytes, or approximately 64,000 TB.

The 64-bit architecture implies the use of 64-bit pointers. The 64-bit long type provides a basic integral arithmetic data type that corresponds to the computational size of hardware registers in 64-bit execution mode. Together, they are referred to as the "LP64" data model. The corresponding model for 32-bit programs is ILP32, that is, the types int, long, and pointer are each 32-bit quantities. In the LP64 model, only long and pointer size is different; other data types, including int, remain the same size as in the ILP32 model. The LP64 model is an industry convention that has been agreed to by system vendors including IBM, Digital, SGI, HP, and Sun.

In the implementation of AIX Version 4.3, 32-bit operations are a native part of the 64-bit processor. Therefore, 32-bit applications that currently run on AIX Version 4 will also run without change. In addition, AIX Version 4.3 runs on new and existing models of the RS/6000 family in 32-bit mode.

The AIX kernel forms the heart of the AIX operating system. The kernel is preemptible, pageable, dynamic, and extendable. AIX V4.2 is a multitasking and multithreading system that uses processes and threads¹. A unique aspect of the AIX kernel is that each user thread has a kernel thread associated with it. The mapping of user threads to the kernel threads is done by using virtual processors. Multiple parts of a program can be running at the same time, even on different processors (parallel processing). AIX provides a processor affinity technique that

¹ A thread is a sequence of instructions that can be scheduled, similar to a process. The POSIX.4a standard defines a process as an address space with one or more threads of control. A thread is designed to be lightweight and inexpensive (in terms of resources consumed) to create, terminate, schedule a thread, or to synchronize with it. AIX allows a maximum of 512 simultaneous user-level threads per process.

tries to run the thread on a processor that it last ran on. This provides for the data to remain in processor cache and optimizes performance.

AIX is well suited to run in a symmetric multiprocessor (SMP) system because of its multithreading kernel. Scalability on SMP systems is high.

One of the unique features of AIX and other UNIX variants is that every file is simply a bit stream and every device is a file. As a result, programs and scripts can be written independent of the type of devices they are going to use. It is very different from an operating system like OS/390 (MVS), which uses file access methods such as VSAM to provide access to disk data. If you wish to use a tape or terminal device instead, you must change your code to support the new devices. In AIX, you simply redirect the output in any of a variety of ways and the code does not have to change.

AIX is also a multiuser system; that is, *multiple concurrent* users are supported. A number of users can be logged into an AIX system at the same time from various types of devices, such as modems, ASCII terminals, and X-stations. The super user is called *root*.

AIX is a cost-effective platform (provided one is satisfied with the level of security and integrity that it provides) for simple workloads for which ample system resources have been configured. The following are the generic strengths of UNIX:

- Cheap hardware and software
- Multiuser, multitasking system
- Large number of packaged applications
- Large skill base
- Networking

C.1.2 Windows NT features

The Windows NT operating system architecture is a hybrid architecture following the client/server and layered models. It is also object-oriented and supports symmetric multiprocessing. Its features include 32-bit addressing, virtual memory support, preemptive multitasking, multithreading, multiprocessing support, platform independence, built-in modular networking, system security, and enhanced system integrity.

The operating system is built on a layered approach similar to the UNIX operating system. The kernel-mode portion of Windows NT is called the Windows NT *executive*. It implements virtual memory management, object (resource) management, I/O and file systems (includes network drivers), interprocess communication, and portions of the security system. These components interact with one another in a modular, layered fashion. The lowest layer is the Hardware Abstraction Layer (HAL) that manipulates the hardware directly. This is a small portion of the code and this is the only part that needs to be changed when porting the operating system from one hardware platform to the other. The layer above the HAL is the *micro kernel*, which communicates only with the Windows NT executive, thus making the “core” of Windows NT very stable.

The non-privileged mode is the user mode. The user-mode processes, like many other system resources, are defined as objects in Windows NT. A process is a data structure that comprises of the following:

- An executable program, which defines initial code and data
- A private address space, which the virtual memory manager (VMM) allocates for the process
- System resources that are allocated to the process
- At least one thread of execution

Other operating systems also use privileged and non-privileged modes. Windows NT is unique with respect to where it draws the line between the two modes. It is sometimes referred to as a micro kernel-based operating system. The idea behind the pure micro kernel concept is that all operating system components except a small core (the micro kernel) execute as user-mode processes.

One of the disadvantages to the pure micro kernel design is slow performance. This is because every interaction between operating system components would require an interprocess message apart from context switching. But, Windows NT takes a modified approach that is between pure micro kernel and monolithic design. The basic operating system subsystems, including the Process Manager and the Virtual Memory Manager, execute in kernel mode, and they can communicate with one another by using function calls for maximum performance.

The Windows NT operating system environments rely on services, known as native API, that the kernel mode exports to carry out tasks that they cannot carry out in user mode. This API is made up of about 250 functions. These functions are accessed through software-exception system calls that is a hardware-assisted way to change execution modes from user mode to kernel mode.

Windows NT is architecturally a single-user operating system. Multiple users can be defined, but only one user at a time can interactively log in. Some may say that Windows NT is a serially reusable multiuser operating system. When connected in a network, Windows NT systems can inter operate in a peer-to-peer relationship called a workgroup or in a domain.

In a workgroup model, each computer maintains its own user account database. Computers can share resources over the network by using shares. Any user can create a share and set permissions to that share. In a domain model, however, a Windows NT Primary Domain Controller maintains a centralized user account database. Users may log in to the domain or any computer that is part of the domain because each computer can still maintain a local user account database.

Windows NT is actually designed to work in a client-server mode. To access a server's resources, you need your own personal computer running, for example, Windows 95/98, Windows for Workgroups, Windows NT Workstation, or Windows NT Server. There is no support for ASCII terminals or Xstations.

C.1.3 OS/390 features

OS/390 was introduced in 1996. It supports the architectural strengths of MVS while expanding the operating system to include e-commerce, Domino technology, Web servers, object-oriented programming language support, systems management, and ERP solutions, while maintaining all the promised qualities of service and superior performance. The future releases will keep supporting object-oriented technology, Enterprise Java Beans, leading edge security technology, enhanced UNIX support, and enhanced performance using

TCP/IP – all technologies necessary to run a successful Web enabled, e-business.

MVS, developed 25 years ago, introduced a secure, reliable, multiprocessing operating system. One of the main reasons OS/390 and its predecessors have been and remain so successful and long-lived is because of the almost fervent attention paid to reliability, availability, and serviceability (RAS) from its earliest days. The design philosophy of MVS is that hardware failures and errors will occur and must be handled, and all single error or hardware failures are recovered by hardware and the operating system.

As the name multiple virtual storage (MVS) implies, each application program (batch or interactive) is given its own address space that can be up to 2 GB in size. The separate address spaces provides a layer of protective insulation between the application programs. Though there is isolation, programs can communicate with each other via the common area (area in central storage common to all programs – includes operating system code) or cross-memory services. In addition to the 2 GB of virtual storage in an application's own address space (primary address space), application programs store and retrieve information stored in many data spaces and hyperspace (ranging in size from 4 KB to 2 GB). Data spaces and hyperspace are used to store data that can be accessed by one or more application programs.

The two major elements of MVS are the *base control program* (BCP), which provides the functions for controlling the resources of the S/390 systems, I/O channels, processor storage, and virtual storage and the *job entry subsystem* (JES), which manages and schedules the flow of work (that is, jobs) to the BCP and manages the outputs of these jobs. Then, there are additional components, such as time sharing option extensions (TSO/E) that allows users to interact directly with the system. The eNetwork Communications Server for OS/390 manages the communication traffic between systems in the network. It supports both systems network architecture (SNA) and TCP/IP communication networks. The LAN Server for MVS (LSFM) enables LAN-based workstation users to gain rapid access to S/390 disk space. Text, graphics, images, video, and sound can be stored on one system and accessed by many workstation users with the network file system (NFS) file serving capabilities of the LSFM.

Users familiar with UNIX commands can use their skills immediately with OS/390 UNIX services (UNIX 95 compliant and soon will be UNIX 98 compliant), formally known as OpenEdition MVS. The portable operating system interface for computer environments (POSIX) provides for a variety of functions for process creation, control, and communication. Effectively, these functions are a subset of the UNIX kernel functions now included in OS/390. They can access the Hierarchical File System (HFS), device- and class-specific functions, input/output primitives, C language-specific services, and data interchange/archive format.

HFS files are byte-oriented, while standard OS/390 files are record-oriented. The HFS allows for long file names in mixed case – up to 1023 bytes for a fully-qualified name. It uses hierarchical directories, treats all data as byte streams, provides utilities for handling the files, provides permission control, supports concurrent writes to the same file from multiple address spaces, and all other features as specified in the POSIX standard. Within OS/390, the HFS is implemented through the Data Facility Storage Management Subsystem (DFSMS), which means that the file system can be automatically backed up and

periodically archived. Security for the HFS is integrated with the resource access control facility (RACF) security services.

Windows NT applications can be ported to OS/390 using Wind/U. This is a cross-platform development tool that enables C and C++ programs written to the Windows NT APIs to be ported to OS/390. The code has to be recompiled and linked with the Wind/U library using IBM development tools on OS/390 to generate a native UNIX version of their application. The Wind/U-ported applications have the same functionality as the original Windows NT programs. This support enables consolidation of Windows NT server workloads on the S/390, thus providing Windows NT-based business applications with scalability, availability, better management, administration, and a secure operating environment.

A unique feature of OS/390 is its WorkLoad Manager (WLM). WLM allows a system administrator to set very general goals for the systems workloads and WLM manages the systems resources and dispatching priorities to try and achieve the goals. WLM still makes use of the system resources manager (SRM) to achieve this. It is more of a queue manager in that it maintains a queue for each of the service classes that are defined to it by the system administrator. This feature is what makes OS/390 a platform of choice for mixed workloads.

The base UNIX operating systems have no mechanism for altering workload mix to optimize resource utilization or to meet service-level requirements. For job scheduling and management, AIX has a job-scheduler that is event driven. In addition, it has a Load Leveler function that is an implementation of a network-based dynamic job routing program.

Windows NT Server operating system was never designed to handle heterogeneous workloads well; the design is optimized to handle a single application type and there are minimal tuning parameters. As a result, Microsoft recommends that each application be handled by one or more servers dedicated to that application. This is partly a legacy of its heritage in desktop solutions, where typically only one item of work s run at a time.

OS/390 system administration functions and OS/390 clustering (parallel sysplex) make it the platform of choice for those who require:

- Mixed workloads
- High-volume transaction processing
- Database access
- Commercial batch
- Operational savings from server consolidation
- High availability
- A secure environment
- Scalability
- Reliability, availability, and serviceability (RAS)
- Data integrity of a centralized, shared database
- Management of a complex or constantly changing environment, such as a service bureau or service provider

Appendix D. OS/390 Net.Commerce development tools

Compiling on OS/390 is generally achieved through the use of a makefile. We provide a sample makefile here which may save you some time when developing and compiling on OS/390.

D.1 OS/390 Net.Commerce sample makefile

Net.Commerce Commands and Overrideable Functions are written in C++. You must compile their C++ source on OS/390 to enable them to run correctly.

You will need to make the following changes:

- Change NC_ROOT to your Net.Commerce installation root.
- Change the DB2 path 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
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)

```

D.2 Precompiling C code containing static SQL with Net.Commerce V3.1.1

For users using an earlier release of Net.Commerce than V3.1.2 we provide a REXX exec that we used for precompiling C code that contained SQL statements (EXEC SQL). You should change the following:

- Change db2subsys to SYSTEM (your DB2 subsystem name).
- Change db2hlq to the high-level qualifier of your DB2 subsystem.
- Change dbrm to the name of your data set where you want the DBRM written to.
- Change dbrmm to the name of the member you want the DBRM written to.

For precompiling C code containing static SQL, customers have also used the esql package available for download on the Web.

```
/* ***REXX Start***** */
/*
/*
/*
/* Exec Name:      DB2PREC
/*
/*
/* Description:  Runs the DB2 precompiler.  This exec will copy
/*              C source from the hierarchical file system
/*              and place it into a temporary MVS data set for
/*              processing by the DB2 precompiler.  The output
/*              from the DB2 precompiler (modified C source) will
/*              be copied back into the hierarchical file system.
/*
/*              Optionally run BIND to bind to a plan.
/*
/*              This exec can be run from the OpenEdition MVS shell.
/*
/* Invocation:    DSNHPC, BIND
/*
/* Variables :    DB2 subsystem name
/*              High level qualifier of DB2 data sets
/*
/* Input parms:
/*
/* infile         (required)  HFS INPUT FILE TO DB2 PRECOMPILER
/* outfile        (required)  HFS OUTPUT FILE FROM DB2 PRECOMPILER
/*
/* path           (optional)  PATH FOR BOTH INPUT and OUTPUT
/* plan           (optional)  NAME OF PLAN IF YOU WANT TO BIND
/*
/* Examples:
/*
/* Run DB2 precompile but don't bind it to a plan
/*
/*   precomp path=/u/user/ infile=db2cpgm2.i outfile=db2cpgm2.c
/*
/* Run DB2 precompile and bind it to a plan called db2cplan
/*
/*   precomp path=/u/user/ infile=db2cpgm2.i outfile=db2cpgm2.c
/*           plan=db2cplan
/*
```

```

/* Run DB2 precompile & bind and have different input and output path */
/*
/*   precomp infile=/u/user/db2cpgm2.i outfile=/u/kjellaa/db2cpgm2.c */
/*       plan=db2cplan */
/*
/*****

Address TSO

/*Trace '?r'*/

Parse Arg arg.1 arg.2 arg.3 arg.4 arg.5

/* free and alloc sysprint and system */

'FREE DDNAME(SYSPRINT) '
'ALLOC DDNAME(SYSPRINT) DA(*) '
'FREE DDNAME(SYSTEM) '
'ALLOC DDNAME(SYSTEM) DA(*) '

/* Define SYSTEM variables */

uid      = sysvar(sysuid)
db2subsys = 'SYSTEM(DDH3) ' /* DB2 subsystem */
db2hlq    = 'DSN510H1' /* DB2 hlq */
syslibd   = '||db2hlq|.SRCLIB.DATA' || ' ' /* DBRM source lib */
dbrm      = '||MY.SCMNDBRM' || ' ' /* DBRM output file */
preinp    = '||uid|.DB2.PREINP' || ' ' /* input work file */
preout    = '||uid|.DB2.PREOUT' || ' ' /* output work file */

/* Define REXX variables */

pathi = ''
patho = ''
pathx = ''
plan  = ''
rch   = 0
rcx   = 0

/* Check the invocation parameters */

Do i = 1 To 5
  If arg.i <> ''
    Then
      Do
        Select
          When SUBSTR(arg.i,1,5) = 'plan='
            Then plan=SUBSTR(arg.i,6)
          When SUBSTR(arg.i,1,7) = 'infile='
            Then
              Do
                pathi=SUBSTR(arg.i,8)
                endname=POS('.',arg.i)
                progname=SUBSTR(arg.i,8,endname-8)
                UPPER progname
              End
          When SUBSTR(arg.i,1,8) = 'outfile='
            Then patho=SUBSTR(arg.i,9)
        End
      End
    End
  End
End

```



```

        When SUBSTR(arg.i,1,5) = 'path='
            Then pathx=SUBSTR(arg.i,6)
        Otherwise
            Do
                Say 'INVALID VALUE. PARAMETER #'||i||'='||arg.i
                Say 'PRECOMP ended with RC = 16'
                Return 16
            End
        End
    End
End

/* Fold plan name to upper case */

Upper plan

/* Test input parameter combinations */

If (pathi = '' | patho = '')
Then
    Do
        Say 'INPUT FILENAME, OUTPUT FILENAME NOT SPECIFIED'
        Say 'PRECOMP ended with RC = 16'
        Return 16
    End
If pathx <> '' & (pathi = '' | patho = '')
Then
    Do
        Say 'PATH SPECIFIED, BUT INPUT AND OUTPUT FILE NOT SPECIFIED'
        Say 'PRECOMP ended with RC = 16'
        Return 16
    End
If pathx <> ''
Then
    Do
        l = LENGTH(pathx)
        If SUBSTR(pathx,1,1) <> '/' | SUBSTR(pathx,l,1) <> '/'
            Then
                Do
                    Say 'PATH SPECIFIED, BUT DOES NOT START AND END WITH /'
                    Say 'PRECOMP ended with RC = 16'
                    Return 16
                End
            End
        End
    End
If pathx <> '' & (POS('/',pathi) <> 0 | POS('/',patho) <> 0)
Then
    Do
        Say 'PATH SPECIFIED, BUT INPUT OR OUTPUT FILE NAME CONTAINS /'
        Say 'PRECOMP ended with RC = 16'
        Return 16
    End
If (pathi = patho)
Then
    Do
        Say 'PRECOMPILE, BUT SAME PATH NAME FOR INPUT AND OUTPUT'
        Say 'PRECOMP ended with RC = 16'
        Return 16
    End
End

```

```

If plan <> ''
Then
Do
plen = LENGTH(plan)
If plen > 8
Then
Do
Say 'PLAN NAME IS TOO LONG, MUST BE 8 CHARACTERS OR LESS'
Say 'PRECOMP ended with RC = 16'
Return 16
End
End

/* Create variables */

If pathx <> ''
Then
pathi = pathx||pathi
If pathi <> ''
Then
Do
pathi = ''||pathi||'' /* HFS input */
End
If pathx <> ''
Then
patho = pathx||patho
If patho <> ''
Then
Do
patho = ''||patho||'' /* HFS output */
End

/* Delete and allocate MVS work data sets */

Do
If sysdsn(preinp) = 'OK'
Then
"delete ("preinp") nonvsam"
If sysdsn(preout) = 'OK'
Then
"delete ("preout") nonvsam"
End

/* Allocate work files */

Do
If sysdsn(preinp) <> 'OK'
Then
'ALLOC FI(PREINP) DA('preinp') UNIT(SYSALLDA) NEW REUSE SPACE(16,16)
TRACKS RECFM(F B) LRECL(80)'
Else
'ALLOC FI(PREINP) DA('preinp') RECFM(F B) LRECL(80) SHR REUSE'
If sysdsn(preout) <> 'ok'
Then
'ALLOC FI(PREOUT) DA('preout') UNIT(SYSALLDA) NEW REUSE SPACE(16,16)
TRACKS RECFM(F B) LRECL(80)'
Else
'ALLOC FI(PREOUT) DA('preout') RECFM(F B) LRECL(80) SHR REUSE'

```

```

End

/* Allocate HFS files */

Do
    'ALLOCATE DDNAME(HFSINP) PATH('pathi') PATHOPTS(ORDWR,OCREAT)',
    'PATHMODE(SIRUSR,SIWUSR)'
End

/* OCOPY from OMVS to MVS */

Do
    'OCOPY INDD(HFSINP) OUTDD(PREINP) TEXT CONVERT(NO)'
    Call Setrch
    'free ddname(preinp)'
    If rcx > 0
        Then
            Return rcx
    End
End

/* Free input files */

'FREE DDNAME(HFSINP)'

/* executing the DB2 precompiler */

If plan = ''
    Then
        Do
            plan = 'dummy' /*dummy name for plan to get through precomp */
        End
    Do
        dbrrmm = '||'|'MY.SCMNDBRM('progrname')'|'
        'ALLOC DDNAME(SYSIN) DA('preinp') SHR REUSE'
        'ALLOC DDNAME(SYSCIN) DA('preout') SHR REUSE'
        'ALLOC DDNAME(SYSLIB) DA('syslibd') SHR REUSE'
        'ALLOC DDNAME(DBRMLIB) DA('dbrrmm') SHR REUSE'
        'ALLOC FI(SYSUT1) UNIT(VIO) NEW REUSE SPACE(16 16) TRACK'
        'ALLOC FI(SYSUT2) UNIT(VIO) NEW REUSE SPACE(16 16) TRACK'
        prog = "'x.y(DSNHPC)'"
        parm = "'HOST(C),MARGINS(1,80),NOOPTIONS,FLAG(W)'"

        Call PGMCALL prog parm
        Call Setrch

        'FREE DDNAME(SYSIN)'
        'FREE DDNAME(SYSCIN)'
        'FREE DDNAME(SYSLIB)'
        'FREE DDNAME(DBRMLIB)'
        If rcx > 4
            Then
                Return rcx
        End
    End

/* executing the DB2 DSN BIND command */

If plan <> 'dummy'
    Then

```

```

Do
    'ALLOC DDNAME (DBRMLIB) DA ('dbrm') SHR REUSE'
    members = 'NETCOMDB,NETCDB2,DTWGAV21,'
    Queue 'BIND PLAN('plan') MEMBER('members||progrname') ACTION(REPLACE)
ISOLATION(CS) CURRENTDATA(NO) FLAG(W) '
    Queue 'RUN PROGRAM(DSNTIAD) PLAN(DSNTIA51) '
    Queue 'End'
    'dsn' db2subsys
    Call Setrch
    'FREE DDNAME (DBRMLIB) '
End

/* executing OCOPY to copy translated file to hfs */

If patho <> ''
Then
    Do
        'FREE DDNAME (PREOUT) '
        'ALLOC DDNAME (MVSINP) DA ('preout') SHR REUSE'
        'ALLOC DDNAME (HFSOUT) PATH ('patho') PATHOPTS (ORDWR,OCREAT) ',
            'pathmode(sirusr,siwusr) '
        'OCOPY INDD (MVSINP) OUTDD (HFSOUT) TEXT CONVERT (NO) '
        Call Setrch
        'free ddname(mvsinp) '
        'free ddname(hfsout) '
        If rcx > 0
            Then
                Return rcx
    End
    Say 'Precomp/bind ended With highest rc =' rch
Return rch

/* pgmcall subroutine for calling a program with a parm field */

Pgmcall:
    Parse Arg pgm a
    a = STRIP(a,"B","'")
    p = LENGTH(a)
    Interpret "p1 = 'D2X(LENGTH(a))'x"
    p1 = OVERLAY(p1,'0000'x,3-LENGTH(p1)) || a
    Parse Var pgm "(" pg ")"
    Address "LINKPGM" PG "p1"
Return rc

/* setrc subroutine for keeping highest return code */

Setrch:
    rcx = rc
    If rcx > rch
        Then
            rch = rcx
Return

```

Appendix E. REXX sample to generate DB2 for OS/390 utility job

These sample programs (execs) will automate part of the cleanup work you have to do after data migration.

E.1 Reorganize and copy REXX sample

The following is a reorganize and copy sample:

```
/*REXX*****  
/*      REXX EXEC USED TO GENERATE REORG/COPY JOBS      */  
/*****  
"ALLOC DD(IN1) DS('HAUSER.SPUFI.TSOUT') SHR REUSE"  
"EXECIO * DISKR IN1 (STEM LISTV."  
IF RC^=0 THEN EXIT  
"ALLOC DD(OUT) DS('HAUSER.REORGTS') NEW"  
IF RC^=0 THEN EXIT  
QUEUE "//REORGTS JOB CLASS=A,MSGCLASS=X,REGION=5M,NOTIFY=&SYSUID "  
DO I=1 TO LISTV.0  
  PARSE VALUE LISTV.I WITH DBTS RESTE  
  K=RIGHT(I,3,0)  
  QUEUE "//R"K" EXEC DSNUPROC,SYSTEM=DB2P,UID='REO"K" "  
  QUEUE "//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(15,15)) "  
  QUEUE "//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(15,15)) "  
  QUEUE "//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(15,15)) "  
  QUEUE "//SORTWK04 DD UNIT=SYSDA,SPACE=(CYL,(15,15)) "  
  QUEUE "//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(15,15)) "  
  QUEUE "//SORTOUT DD UNIT=SYSDA,SPACE=(CYL,(15,15)) "  
  QUEUE "//SYSPRINT DD SYSOUT=*" "  
  QUEUE "//SYSCOPY DD DSN=MIGR.COPY"K",DISP=(NEW,CATLG,DELETE), "  
  QUEUE "// UNIT=SYSDA,SPACE=(CYL,(15,15),RLSE), "  
  QUEUE "// DCB=(RECFM=FB,LRECL=4096,BLKSIZE=24576,BUFNO=30) "  
  QUEUE "//SYSREC DD DSN=MIGR.RECS"K",DISP=(NEW,DELETE,KEEP), "  
  QUEUE "// UNIT=SYSDA,SPACE=(CYL,(15,15),RLSE), "  
  QUEUE "// DCB=(RECFM=FB,LRECL=4096,BLKSIZE=24576,BUFNO=30) "  
  QUEUE "//SYSIN DD * "  
  QUEUE " REORG TABLESPACE "DBTS" LOG NO "  
  QUEUE " SORTDATA SORTKEYS COPYDDN(SYSCOPY) "  
"EXECIO " QUEUED() " DISKW OUT"  
END  
"EXECIO 0 DISKW OUT (FINIS"  
"FREE DD(OUT) "  
"EXECIO 0 DISKW IN1 (FINIS"  
"FREE DD(IN1) "  
EXIT
```

E.2 RUNSTATS REXX sample

The following is a RUNSTATS REXX sample:

```
/*REXX*****  
/*      REXX EXEC USED TO GENERATE RUNSTATS JOBS      */  
/*****  
"ALLOC DD(IN1) DS('HAUSER.SPUFI.TSOUT') SHR REUSE"  
"EXECIO * DISKR IN1 (STEM LISTV."  
IF RC^=0 THEN EXIT
```

```

"ALLOC DD(OUT) DS('HAUSER.STATGEN') NEW"
IF RC^=0 THEN EXIT
QUEUE "//STATS JOB CLASS=A,MSGCLASS=X,REGION=5M,NOTIFY=&SYSUID "
DO I=1 TO LISTV.0
    PARSE VALUE LISTV.I WITH DBTS RESTE
    FICNO=DATE('J')
    K=RIGHT(I,3,0)
        QUEUE "//E"K" EXEC DSNUPROC,SYSTEM=DB2P,UID='STAT"K"' "
        QUEUE "//SYSPRINT DD SYSOUT=* "
        QUEUE "//SYSIN DD * "
        QUEUE " RUNSTATS TABLESPACE "DBTS" TABLE INDEX "
    "EXECIO " QUEUED() " DISKW OUT"
END
"EXECIO 0 DISKW OUT (FINIS"
"FREE DD(OUT) "
"EXECIO 0 DISKW IN1 (FINIS"
"FREE DD(IN1) "
EXIT

```

Appendix F. Alternative approach to database migration

Once the database for the BASEMALL was created and loaded on OS/390, we proceeded to load the rest of the tables in the database using the *import* and *loadtbls* utilities. The order for the loading was determined from the data models for the various functional groups documented in the online help. The order that we followed is as follows:

STRCGRY, MERCHANT, SHIPMODE, MSHIPMODE, PRSPCODE,
SHIPPING, PRODUCT, SHOPGRP, PRODPRCS, CATEGORY, CGRYREL,
CGPRREL, CATESGP, PRODSGP, PRODATR, PRODDSTATR, SHOPPER,
SHOPDEM, SHADDR, ACCTRL, MCUSTINFO, MERCHANTTAX,
SG_STORES, SHOPPINGS, ORDERS, ORDERPAY, ORDPAYMTHD,
SHIPTO, ACC_GROUP, ACC_USRGRP, ACC_CMDGRP, CMDS,
ACC_MODE, MACROS

The above tables were not all empty. The SHOPPER, SHOPDEM, SHADDR, ACCTRL, ACC_GROUP, ACC_USRGRP, ACC_CMDGRP, CMDS, ACC_MODE, and MACROS tables had data that was not updated. The extra data introduced by the application was inserted into these tables.

For the tables that had extra columns on the OS/390 version of Net.Commerce, the extra columns (as specified in Table 5) were added to the tables on NT and then the data was exported for upload into the tables on OS/390.

Table 5. Tables with extra columns

Table Name	Column(s)	Data Type
MERCHANT	MERUID	CHAR(36)
SHOPPER	SHPPREFERREDCURR	CHAR(3)
SHOPGRP	SGOID	CHAR(36)
SHIPMODE	SMTRKNAME SMTRKURL SMTRKSH SMTRKSP SMTRKICON SMTRKTYPE	VARCHAR(64) VARCHAR(64) VARCHAR(64) INTEGER VARCHAR(64) CHAR(8)
MSHIPMODE	MMTRKSPNBR	VARCHAR(64)
SHIPPING	SPCURR	CHAR(3)
CATEGORY	CGOID	CHAR(36)
PRODUCT	PROID	CHAR(36)
PRODPRCS	PPOID	CHAR(36)
PRODATR	PATYPE PAOID	INTEGER CHAR(36)
PRODDSTATR	PDROID	CHAR(36)
PRODSGP	PSTOID	CHAR(36)

Table Name	Column(s)	Data Type
SHIPTO	STBASEPRICE STBASECURR STTRKNBR STTRKDATE	NUMERIC(15,2) CHAR(3) VARCHAR(64) TIMESTAMP
CATESGP	CSTOID	CHAR(36)
DISCCALC	DCCSCLCURR	CHAR(3)
ACC_USRGRP	MER_RFNBR	INTEGER

The data for the MACROS, CATESGP, and PRODSGP tables contained information about the relative path location of macros within the directory structure. This path information contained backslashes for the tables on NT that had to be changed to forward slashes for use on OS/390. The data for these tables was exported to ASCII delimited files and converted to EBCDIC (using ftp), then edited to change the slashes and loaded into the OS/390 tables using the loadtbls utility. The extra rows in the MACROS table that belonged to the application were inserted using SQL inserts.

The KEYS table needed special handling. We found that the KEYS table contains the current maximum values of the primary keys of the following tables:

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

The Net.Commerce system uses this information to set primary keys for new rows. If you populate the database without using Net.Commerce administrator, you must update this table manually to include the highest keys you use for each table. The highest keys for each table were found by using the following SQL:

```
select max(primarykey) from <database_owner>.table;
```

The KEYS table was then updated with the example SQL update command:

```
update <database_owner>.keys set KEYMAXID=29153  
  
where KEYRFNBR=1;
```

The application we chose to port did not have any extra Overrideable Functions, so the OFS table did not need to be updated for these. The POOLS, POOL_CMD, and TASKS tables also did not need to be updated.

Now that we identified all the tables owned by Net.Commerce, we had to identify tables that were unique to the application. These tables are identified from the Control Center of DB2 for NT. The DDL for these tables was extracted from the DB2LOOK output and executed on OS/390 to create these tables. After creating these extra tables, the data was loaded into the OS/390 DB2 database using import from the Windows NT DB2.

There are some platform-specific differences that one has to be aware of. They are:

- Every primary key that is created for a table on DB2 for OS/390 has to have a unique index associated with it. The command for creating a unique index looks like this:

```
CREATE UNIQUE INDEX "<index_name>"
ON "<database_owner>". "<table_name>"
("column_name", ["column_name"],...);
```

- The command to create foreign keys on DB2 for NT looks like this:

```
ALTER TABLE "SA"          ". "ACC_CMDGRP"
ADD CONSTRAINT "FK_ACCCMDS" FOREIGN KEY
("CMD_REFNUM")
REFERENCES "SA"          ". "CMDS"
("REFNUM")
ON DELETE CASCADE
ON UPDATE NO ACTION;
```

This would not work on DB2 for OS/390. This would have to be changed to:

```
ALTER TABLE "SA"          ". "ACC_CMDGRP"
ADD FOREIGN KEY "FK_ACCCMDS"
("CMD_REFNUM")
REFERENCES "SA"          ". "CMDS"
("REFNUM")
ON DELETE CASCADE;
```

F.1 DB2 housekeeping

For a detailed description of how to perform database housekeeping after the migration of the data, please refer to 4.1.7, "Database housekeeping" on page 73.

Appendix G. Special notices

This publication is intended to help project leaders and technical specialists who are working with Net.Commerce understand the tasks involved with the migration of Net.Commerce for Windows NT or UNIX to Net.Commerce for OS/390. The information in this publication is not intended as the specification of any programming interfaces that are provided by Net.Commerce for Windows NT, UNIX, or 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 them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating

environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

This document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AS/400
CICS	DATABASE 2
DB2	DFSMS
DRDA	eNetwork
IBM	IMS
Language Environment	MQ
MQSeries	Net.Data
Netfinity	OpenEdition
OS/390	OS/400
Parallel Sysplex	PS/2
RACF	RS/6000
S/390	System/390
VisualAge	VM/ESA
VSE/ESA	VTAM
WebSphere	400

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

SET and the SET logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Appendix H. Related publications

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

H.1 IBM Redbooks publications

For information on ordering these publications see “How to get IBM Redbooks” on page 117.

- *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

H.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 Format)	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

H.3 Other resources

These publications are also relevant as further information sources:

- *IBM Commerce Integrator User's Guide for AIX*, SC09-2878
- *IBM Commerce Integrator User's Guide for Sun Solaris*, SC09-2879
- *IBM Commerce Integrator User's Guide for Windows NT*, SC09-2865
- *IBM Net.Commerce for AIX Installing and Getting Started*, GC09-2627
- *IBM Net.Commerce for OS/390 Configuring and Getting Started*, GC24-5862
- *IBM Net.Commerce for Windows NT Installation and Getting Started*, GC09-2626
- *Porting a UNIX Application to OS/390 UNIX: Problems Encountered and Lessons Learned* (Available at:
<http://www.s390.ibm.com/oe/aixdb2/aixdb2.html>)
- *Porting Applications to the OpenEdition OS/390 Platform*, GG24-4473
- *Program Directory for Net.Commerce for OS/390*, GI10-4657

- *Program Directory for Net.Data for OS/390 with National Language Features*, GI10-6971 (Must be ordered with software product)
- *WebSphere Application Server for OS/390 HTTP Server Planning, Installing, and Using*, SC31-8690 (Available at: <http://www.ibm.com/s390/os390>)

H.4 Referenced Web sites

These Web sites are also relevant as further information sources:

- <http://www.download.com>.
- <http://enterprise.torolab.ibm.com/~rintjema/netcom3/refs/srimfile.htm>.
- <http://www.ibm.com/s390/os390>
- <http://www.ibm.com/e-business/emarkinfo/>
- <http://www.ibm.com/software/data/ims/imswwwc.html>
- <http://www.ibm.com/software/data/net.data/library.html>
- <http://www.s390.ibm.com/nc/ecommerce/doc.html#net>
- <http://www.s390.ibm.com/nc/ecommerce/sampcode.html>
- <http://www.s390.ibm.com/oe/aixdb2/aixdb2.html>
- <http://www.s390.ibm.com/oe/bpxalpor.htm>
- <http://www.s390.ibm.com/oe/bpxaltoy.html>
- <http://www.s390.ibm.com/products/oe/bpxalp03.html>

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.ibm.link.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.ibm.link.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.ibm.link.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
-------	--------------	----------

[illegible]

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

- AIX.** Advanced Interactive eXecutive.
- ANSI.** American National Standards Institute.
- ASCII.** American National Code for Information Interchange.
- API.** Application Programming Interface.
- BCP.** Base Control Program.
- CCF.** Common Connector Framework.
- CGI.** Common Gateway Interface.
- CICS.** Customer Information Control System.
- CLI.** Call Level Interface.
- CLP.** Command Line Processor (DB2).
- CSP.** Commercial Service Provider.
- DB2.** IBM DATABASE 2.
- DBMS.** Database Management System.
- DBRM.** Database Request Module.
- DES.** Digital Encryption Standard.
- DDL.** Data Definition Language.
- DFSMS.** Data Facility Storage Management Subsystem.
- DLL.** Dynamic Link Library.
- DMZ.** De-Militarized Zone.
- DNS.** Domain Name Server.
- DRDA.** Distributed Relational Database Architecture.
- EBCDIC.** Extended Binary Coded Decimal Interchange Code.
- ECI.** External Call Interface.
- EDI.** Electronic Data Interchange.
- EPI.** External Presentation Interface.
- ERP.** Enterprise Resource Planning.
- FTP.** File Transfer Protocol.
- GIF.** Graphic Interchange Format.
- GUI.** Graphical User Interface.
- HAL.** Hardware Abstraction Layer.
- HFS.** Hierarchical File System.
- HTML.** Hypertext Markup Language.
- HTTP.** Hypertext Transfer Protocol.
- IBM.** International Business Machines.
- IDEA.** International Data Encryption Algorithm.
- IEEE.** Institute of Electrical and Electronics Engineers.
- IMS.** Information Management System.
- ISO.** International Standards Organization.
- ISV.** Independent Solution Vendors.
- ITSO.** International Technical Support Organization.
- JCL.** Job Control Language.
- JDBC.** Java Database Connectivity.
- JES.** Job Entry Subsystem.
- JPEG.** Joint Photographic Experts Group.
- LAN.** Local Area Network.
- LE.** Language Environment.
- LPAR.** Logically Partioned Mode.
- LU.** Logical Unit.
- MIME.** Multipurpose Internet Mail Extensions.
- MQ.** Message Queuing.
- MQHRF.** MQSI Rules/Format Header.
- MQI.** Message Queue Interface.
- MQM.** MQSeries Queue Manager.
- MQMD.** MQ Message Descriptor.
- MQSI.** MQSeries Integrator.
- MVS.** Multiple Virtual Storage.
- MRO.** Multi-Region Option.
- NFS.** Network File System.
- ODBC.** Open Database Connectivity.
- OF.** Overrideable Function.
- PC/IXF.** Intergrated Exchange Format, PC Version.
- PDF.** Portable Document Format.
- POSIX.** Portable Operating System Interface.
- RACF.** Resource Access Control Facility.
- RAS.** Reliability, Availability, Serviceability.
- RDB.** Relational Database.
- RDBMS.** Relational Database Management System.
- RI.** Referential Integrity.
- SET.** Secure Electronic Transactions.
- SDSF.** System Display and Search Facility.
- SFS.** Structured File Server.
- S-HTTP.** Secure Hypertext Transfer Protocol.
- SMIT.** System Management Interface Tool.
- SMP.** Symmetric Multiprocessor.
- SMP/E.** System Modification Program/Extended.
- SMTP.** Simple Mail Transfer Protocol.

SNA. Systems Network Architecture.
SQL. Structured Query Language.
SRM. System Resources Manager.
SSL. Secure Sockets Layer.
TCP/IP. Transmission Control Protocol/Internet Protocol.
TSO/E. Time Sharing Option Extensions.
UDB. Universal Database.
URL. Uniform Resource Locator.
VMM. Virtual Memory Manager.
VRML. Virtual Reality Modeling Language.
WLM. WorkLoad Manager.
WWW. World Wide Web.
XID. Exchange Identifier.

Index

Numerics

32 K page 62
4 K page 62
64-bit architecture 92

A

ACC 52
access control 42, 51
AIX 1, 33, 92
 operation 45
 Version 4.3 92
ANSI 79
APF-authorized 30
API 26, 94
applet 33
application 19, 61
application developer 77
architectures 91
AS/400 19
ASCII 19, 20, 93
ASCII to EBCDIC 19
Asynchronous server 26
authentication 87
Availability 10

B

backend 2
 integration 26
Base control program (BCP) 95
BASEMALL 51, 107
basic store 3
batch 95
BLOB 57
BP0 64
browsing 87
buffer pool 64
business logic 2
Business-to-Business 38
buying 87

C

C 20
C++ 19, 77, 80, 81, 96, 97
C/C++ 28
 compiler 20
Catalog Architect 25
categories 51
CategoryDisplay 42
CCF 28
CD-ROM 32
Character 20
CICS 4, 26, 28
 integration 26
CICS integration 26
client/server 9

 model 92
CLP 69
CMD5 31, 51
 table 31
 task 52
CMNCONF 25, 36, 37, 38, 39, 40, 41, 43
CMNCONF screen 37
CMOS 10
COBOL 26, 28
columns 50
commands 7, 19, 20, 28
Commerce Integrator 27
commerce server 9
Common Connector Framework (CCF) 28
Common Gateway Interface (CGI) 27
common tables 62
communication 95
comparison 12, 89
competitor 10
compiling 97
configuration 35
consistency 71
constraints 50
costs 10
CREATE INDEX 67
CREATE TABLE 67
customers 51
customization 3

D

daemon 2
Data 9
Data Definition Language 65
Data Facility Storage Management Subsystem 95
Data Propagator 51
data structures 49
data transformation 26
Database 7
 schema 49
 server 14
Database management system (DBMS) 49
datasets 30
DB2 2, 14, 35, 40, 46, 49, 57, 81, 87, 99, 108
 alter 51
 check data utility 73
 check utility 51, 71
 connect 58
 Control Center 52
 copy utility 51, 71, 74
 design 64
 Estimator 67
 export 51, 52, 57
 housekeeping 51, 73
 import 51, 52, 58
 REORG utility 51
 runstats utility 51, 74
 tables 24

- DB2 Command Line Processor (CLP) 69
- DB2 data definition language (DDL) 50
- DB2 for OS/390 51, 54, 57
 - buffer pool 65
 - database design 64
 - databases 66
 - load utility 69
 - stogroups 64, 66
 - table row 64
 - tablespaces 64, 66
 - Version 6 70
- DB2 for Windows NT 50, 66
- DB2 migration overview 50
- DB2LOOK 50, 51, 57, 66, 108
 - syntax 66
- db2www 40, 42
- db2www.ini 41
- DBMS 49, 65
- DBRM 81
- DD statements 70
- DDDEFS 47
- DDL 50, 57, 65, 66, 108
- delimited ASCII (DEL) 68
- DEMOMALL 38, 40, 51, 52, 56, 61
- development tools 97
- DFSMS 95
- differences 9, 12
- DLL 80
- dll 31
- Domain Name Server 16
- Domino Go Web server 3, 35
- DOS 66
- DRDA 14, 52, 57, 68
- DSNAOCLI 81
- DSNTIAUL 51, 68, 69, 70, 71
- dynamic link libraries 29, 77

E

- EBCDIC 19, 20, 21, 108
- e-business 7
- e-commerce 10
- EDI integration 26
- Education 1
- Electronic data interchange (EDI) 26
- Emergency Response Team 7
- Enterprise Java Beans 94
- Enterprise Resource Planning (ERP) 28
- envvars 40
- ERP 28, 94
- Error Task 32
- Euromall 38
- EXEC 44
- EXEC SQL 99
- ExecDB2Macro 31
- ExecMacro 21
- executables 29
- export 68
- exported 107

F

- Fast Ethernet 88
- File transfer protocol (FTP) 20
- Firewall 13
- Flexibility 10
- FMIDs 33
- ftp 20, 108
- Fulfilment Systems 9
- functions 19, 50, 89

G

- GIF 24
- Grocery 38

H

- HAL 93
- Hardware Abstraction Layer (HAL) 93
- HFS 16, 21, 30, 38, 42, 47, 79, 95
- Hierarchical File System (HFS) 20
- hiperpool 65
- HTML 2, 7, 16, 20, 21, 24, 25, 26, 40, 41, 45, 77
- html root 41
- HTTP 2, 15, 32
 - server 3
- httpd.conf 25, 40

I

- IBM Data Joiner 52
- IBM e-business mark 7
- IBM Global Services 6
- image 24
- images 7, 16
- import 51, 68, 107
- IMS 4, 26, 28
 - connectors 27
 - integration 26, 27
- INCLUDE_PATH 41
- infrastructure 14
- installation 32
- installation and configuration on AIX 33
- installation and configuration on OS/390 34
- installation and configuration on Windows NT 34
- installation directories 45
- instance 33, 35, 40, 42
- IP 2, 10
- ISHELL 42
- ISPF 21, 36, 46, 77, 79
- items 51

J

- Java 28, 33
- Java Database Connectivity (JDBC) 33
- JCL 43, 45, 70
- JDBC 33
- JES 95

K

KEYS 51, 108

L

LANGLVL 80
Language Environment 80, 97
Layered Model 91
LEPARM 44
LIBASCII 19
LIBPATH 29, 30
Load utility 73
LOADTBLS 73
loadtbls 73, 107
locate added columns 55, 60
locate changed tables 54, 59
locate changed views 54, 59
locate modified columns 56, 61
locate new tables 52, 58
locate new views 52, 58
logical tables 64
LONGVAR 62, 67
LPA 30
LPAR 16, 27, 47

M

MACRO_PATH 41
MACROS 32, 50, 51
 table 32
make 80
makefile 30, 77, 80, 97
manageability 10, 64
manipulate 49
MAX(REFNUM) 31
MAX(TKRFBNR) 32
merchants 51
micro kernel 93
micro kernel model 91
Microsoft Visual C++ 77
migrate data 68
migrating the database 49
migration 3, 5, 49, 61
 considerations 19
Monolithic Model 91
MQSeries 4, 26, 28
MQSeries integration 26
Multiple virtual storage (MVS) 95
multiprocessing 93
Multi-region option (MRO) 27
multitasking 93
multithreading 93

N

ncdata_file 36
ncommerce.conf 40
Net.Commerce 1, 2, 6, 9, 13, 41, 50, 59
 architecture 3
 instances 10
 issues 20

Net.Commerce database 19
Net.Commerce for OS/390 16
Net.Commerce HFS hierarchy 43
Net.Commerce Mass Import 51
Net.Data 2, 3, 7, 16, 21, 35
 compatibility 21
 DTW_ODBC 22
 DTW_SQL 22
 functional comparison 23
 macros 21, 23, 41, 77
 path names 22
 special characters 23
Net.Dispatcher 10
network computing 9
Network File System (NFS) 95
NFS 95

O

OFS 31, 32, 51
 table 31
OMVS 20, 42, 78
online help 49
orders 51
OS/390 1, 3, 6, 14, 19, 20, 31, 32, 50, 62, 92, 94, 96, 108
 operation 45
 Security Server (RACF) 10
 testing environment 16
 UNIX System Services 10
 UNIX Systems Services 24
OS/400 1
Overrideable Functions 2, 28, 29, 77, 97, 108

P

PARM 44
Pass directives 25, 40
password 21
pax 20
payment 33
Payment Server 26, 33, 53, 58
PC/IXF 68
performance 10, 11, 69
physical disk volume 64
physical tablespaces 64
platform 9
 choice 9
 choices 11
 sizing 11
POOL_CMD 51
POOLS 51
ported.d2w 21
porting 65
POSIX 95
precompiling 99
price discounting 51
private tablespace 64
PROC 43, 44
Process Task 31
Product Advisor 25, 33, 35, 53, 58

- ProductDisplay 42
- production 6
- products 51
- Project Management 3
- Project Manager 1
- Project Plan 7
- projects 3

R

- RACF 38, 96
- RAS 95
- RDB 68
- recompiled 29
- reference number (REFNUM) 31
- references 58
- referential constraints 50, 69
- referential integrity checking 69
- referential integrity structures 50
- REFNUM 31
- Relational Database (RDB) 52
- Resource Access Control Facility 96
- REXX 28, 74, 99
- root 45, 93
- RS/6000 92

S

- S/390 9
 - platform 9
- Scalability 10
- schema 35
- schema differences 50
- schemas 56
- SCMNSAMP 38, 43, 46
- SDSF 46
- Secure Electronic Transactions (SET) 33
- security 7, 10
- Security Server 4
- sed command 22
- Server Controller 40, 43
- servers 12
- servlet configuration 35
- SET 33
- Setup 52
- shared object code 29
- shared objects (so) 29
- shell commands 30
- shipping 51
- shopper 51
 - groups 51
 - password porting 21
- shopping 87
- skills 4, 9, 85, 95
- SMP/E 32, 47
- sockets 19
- solution 12, 85
- Solution Testing 7
- Solutions Assurance 5, 85
- solutions roadmap 85
- SQL 31, 38, 52, 68, 99, 108

- DDL 65
- insert 69
- inserts 108
- join 54
- join queries 52
- outer join 60

- SRM 96
- SSL 6, 32, 87
- Staging Environments 15
- STEPLIB 30
- stores 51
- stress test 5, 88
- stress testing 7, 85
- Sun Solaris 1
- Symmetric Multiprocessor (SMP) 93
- SYSIBM.SYSCOLUMNS 52, 56, 61
- SYSIBM.SYSTABLES 52, 56, 58
- SYSIBM.SYSYSTABLES 54
- SYSREC 70
- SYSREC00 70
- system management 49
- System Modification Program/Extended (SMP/E) 32

T

- tables 49, 50, 51
- tablespace 64
- tablespaces 73
- tar 20
- TASK_MER_OF 32, 51
- TASK_MER_OF table 32
- TASKS 32, 51
- tasks 7, 28
 - table 32
- tax calculations 51
- TAXPRCODE 52
- TCP/IP 20, 35, 95
- technology 7
- Template Designer 40
- testing environment 15
- TKRFNBR 32
- TSO/E 78, 95
 - commands 20
- TSO/ISPF 33
- tuning 64

U

- UNIX 1, 3, 11, 13, 15, 19, 31, 35, 77, 93, 94, 96
- UNIX System Services 4, 77, 78
- URL 2
- users 51
- USS 30

V

- vi editor 77, 79
- View Task 31
- views 68
- Virtual Memory Manager (VMM) 94
- VisualAge 28

VM/ESA 16
VMM 94
VSAM 64, 93
 datasets 64

W

Web 95
Web server 7, 13, 35, 45, 46
 configuration 25
Web Site Security Scan 7
WebSphere 21, 28
 Visual Studio 77
Windows NT 1, 3, 11, 13, 15, 31, 32, 54, 80, 93, 94
 operation 45
WLM 96
workstation 20

X

X-stations 93

IBM Redbooks evaluation

Migrating Net.Commerce Applications to OS/390
SG24-5438-00

Your feedback is very important to help us maintain the quality of IBM Redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at <http://www.redbooks.ibm.com/>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?

☐ **Customer** ☐ **Business Partner** ☐ **Solution Developer** ☐ **IBM employee**
☐ **None of the above**

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

What other Redbooks would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

SG24-5438-00

Printed in the U.S.A.

