UNISYS

TPC Benchmark™ C Full Disclosure Report

Unisys Corporation
Open Business Systems

Aquanta HS/6 Server

using

Microsoft NT Server 4.0 Enterprise Edition

and

Microsoft SQL Server 6.5 Enterprise Edition

Second Edition
December 19th 1997

Unisys Part Number 4491 3580-100

Second Edition - December 1997

Unisys Corporation believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. Unisys Corporation assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, Unisys Corporation and Microsoft Corporation provide no warranty on the pricing information in this document.

Benchmark results are highly dependent upon workload, specific application requirements, and systems design and implementation. Relative system performance will vary as a result of these and other factors. Therefore, TPC Benchmark C should not be used as a substitute for a specific customer application benchmark when critical capacity planning and/or product evaluation decisions are contemplated.

All performance data contained in this report was obtained in a rigorously controlled environment, and therefore results obtained in other operating environments may vary significantly. Unisys Corporation and Microsoft Corporation do not warrant or represent that a user can or will achieve similar performance expressed in transactions per minute (tpmC) or normalized price/performance ($/tpmC). No warranty of system performance or price/performance is expressed or implied in this report.

Copyright © 1997 Unisys Corporation.

All Rights Reserved. Permission is hereby granted to reproduce this document in whole or in part provided the copyright notice printed above is set forth in full text on the title page of each item reproduced.

Printed in USA, December 1997.

Unisys Corporation Part Number: 4491 3580-100

Unisys and Aquanta are registered trademarks of Unisys Corporation.
Intel, Pentium and Pentium Pro are registered trademarks of Intel Corporation.
Microsoft Windows NT and SQL Server are registered trademarks of Microsoft Corporation.
BEA and Tuxedo are registered trademarks of BEA Systems, Inc.

TPC Benchmark, TPC-C and tpmC are trademarks of the Transaction Processing Performance Council.

Other product names used in this document may be trademarks and/or registered trademarks of their respective companies.

Page Status

Page	Issue
i through xii	-100
0-1 through 0-3	-100
0-4	Blank
1-1 through 1-1	-100
1-2	Blank
2-1 through 2-2	-100
3-1 through 3-3	-100
3-4	Blank
4-1 through 4-5	-100
4-6	Blank
5-1 through 5-8	-100
6-1 through 6-2	-100
7-1 through 7-2	-100
8-1 through 8-1	-100
8-2	Blank
9-1 through 9-3	-100
9-4	Blank
A-1 through A-51	-100
A-52	Blank
B-1 through B-47	-100
C-1 through C-25	-100
D-1 through D-3	-100
D-4	Blank
E-1 through E-2	-100
F-1 through F-8	-100

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678-xyz) indicates the document level. The first digit of the suffix (x) designates a revision level; the second digit (y) designates an update level. For example, the first release of a document has a suffix of -000. A suffix of -130 designates the third update to revision 1. The third digit (z) is used to indicate an errata for a particular level and is not reflected in the page status summary.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark C (TPC-C) conducted on the Unisys Corporation Aquanta HS/6 server. The operating system on the server was Microsoft Windows NT Server 4.0 Enterprise Edition. The DBMS used was Microsoft SQL Server 6.5 Enterprise Edition. The operating system on the clients was Microsoft Windows NT Server 4.0 SP2. The clients ran Microsoft's Internet Information Server 3.0 and Tuxedo 6.3 CFS for NT.

TPC Benchmark Metrics

The standard TPC Benchmark C metrics, tpmC (transactions per minute), price per tpmC (five year capital cost per measured tpmC), and the availability date are reported as required by the benchmark specification.

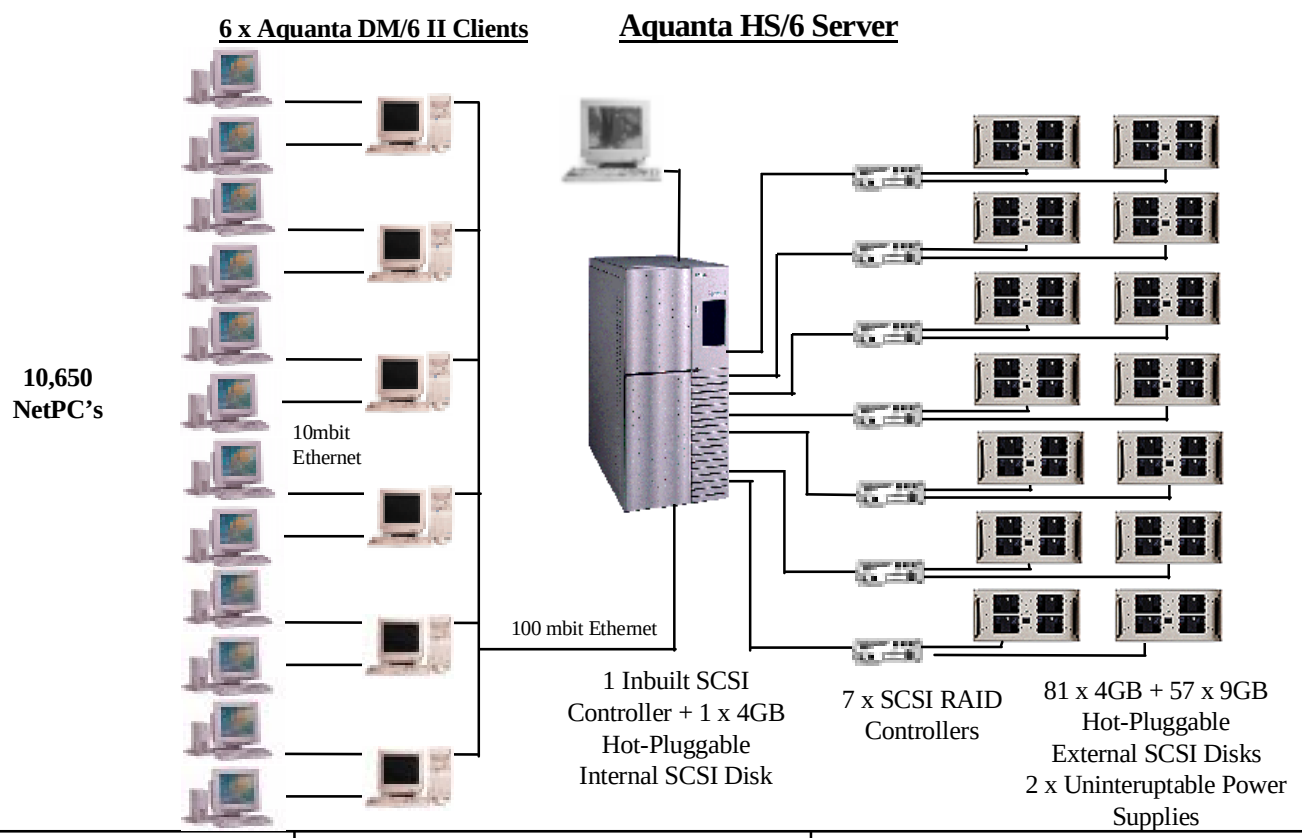
Executive Summary

The following pages contain the executive summary results of the benchmark.

Auditor

The benchmark configuration, environment, and methodology used to produce and validate the test results, along with the pricing model used to calculate the cost per tpmC, were audited by Richard Gimarc of Performance Metrics, Inc. to verify compliance with the relevant TPC specification.

UNISYS	Aquanta HS/6 Server C/S			TPC-C Rev. 3.3 Report Date: 19-Dec-1997
Total System Cost	TPC-C Throughput	Price/Performance		Availability Date
$496,827	**13,089.30 tpmC**	**$37.96 per tpmC**		**30-Jan-1998**
Processors	Database Manager	Operating System	Other Software	Number of Users
6 Pentium® Pro 200 MHz 1MB L2 cache	Microsoft SQL Server 6.5 Enterprise Edition	Microsoft NT Server 4.0 Enterprise Edition	Microsoft IIS 3.0 Tuxedo 6.3 CFS	**10,650**



System Components	Server		Clients	
	Quantity	Type	Quantity	Type
Processors	6	200 MHz Pentium® Pro with 1MB Level 2 Cache	6	200MHz Pentium® Pro with 256KB Level 2 Cache
Memory	1	4096MB	6	256MB
Disk Controllers	7 1	SCSI RAID Inbuilt SCSI	6	Inbuilt IDE
Disk Drives	82 57	4GB 9GB	6	1.5GB
Total Storage		813.47 GB		7.5GB
CD-ROM / Tape	1	CD-ROM Drive	6	CD-ROM Drive

Unisys Corporation	Aquanta HS/6 Client/Server					TPC-C Rev 3.3 19-Dec-1997
Description	Style	Third Party Brand Pricing		Unit Price	Qty.	Extended Price	5 Years Maint.
Server Hardware							
SYS: Aquanta HS/6, w/ CDRom, 0 Proc, 0MB Mem	HS6000122-BAS		1	$9,985	1	$9,985	$864
PROC:3x200MHz PPro/1MB + CPU Riser card	HTC6200-1MB		1	$13,630	1	$13,630	$2,832
PROC:3x200MHz PentiumPro/1MB Cache	HTX6200-1MB		1	$13,306	1	$13,306	$2,760
MEM:ECC Memory Board, 0MB Mem	MEM641-DIM		1	$395	1	$395	$72
MEM:256 MB Memory Upgrade	DIM572-256		1	$2,048	16	$32,768	$5,376
CTRL:RAID Tri-SCSI-2 Ultra PCI	RAD3162-PCI		1	$2,074	7	$14,518	$2,856
CTRL: VGA, 16-bit ISA	VID11-ISA		1	$106	1	$106	
ETHERNET: 100Mbit/sec, PCI 32-bit	SF1001-FET		1	$290	1	$290	$48
CDROM: 16 Speed	CDR1200-SI		1	$178	1	$178	$120
ACC: 6 SCA Drive Cage	CAG61-ADV		1	$425	1	$425	$120
DRW: Disk Drive Drawer, w/ cables, 7U	P/N 11910190	ALR	2	$1,795	16	$28,720	$6,896
DISK: 4GB Drive + 10% spares	HDS4000-WC7		1	$743	91	$67,613	spared
DISK: 9GB Drive + 10% spares	HDS9000-WC7		1	$1,495	54	$80,730	spared
DISK: 9GB Drive + 10% spares	HDS9000-H10		1	$1,602	10	$16,020	spared
MONITOR:14-inch Color	EVG142-COL		1	$219	1	$219	
KEYBD: 104 Key Spacesaver	PCK104-SKB		1	$34	1	$34	
MOUSE: 2 Button PS2	PWM1-PS2		1	$25	1	$25	
PWR: 2200VA UPS, 4U	UPD22001-SXR		1	$2,380	2	$4,760	$1,248
CAB: Rack Cabinet, w/ fill pnls, 36U	CAB361-SXR		1	$1,530	3	$4,590	
CAB: Link kit for 36U cabinets	LNK361-SXR		1	$255	2	$510	
CAB: Bezel kit 36U	BEZ3611-CAB		1	$170	3	$510	
CAB: Stabilizer kit 0U	WGT39581-SXR		1	$120	3	$360	
PNL: L&R side panels 36U	PAN3621-SXR		1	$213	1	$213	
					Subtotal	**$289,905**	**$23,192**
Server Software							
Microsoft NT Server 4.0 Enterprise Edition, incl 25 CALs		Microsoft	3	$3,999	1	$3,999	$0
Microsoft SQL Server 6.5 Enterprise Edtion, incl 25 CALs		Microsoft	3	$28,999	1	$28,999	$10,475
					Subtotal	**$32,998**	**$10,475**
Client Hardware							
SYS: Aquanta DM/6 II, 0 Proc, 0MB Mem	CMT60072-ZFA		1	$896	6	$5,376	$2,592
PROC:1x200MHz PentiumPro/256KB Cache	PRC6200-256		1	$726	6	$4,356	
MEM: 64 MB Memory Upgrade	MTP6-64M		1	$634	24	$15,216	$5,184
CTRL: VGA, 64-bit with 2MB	PCV102-PCI		1	$123	6	$738	
DISK: 1.6GB IDE 3.5 Internal	HDI1600-5		1	$219	6	$1,314	$1,008
CDROM: Twelve Speed IDE	CDR1200-AI		1	$126	6	$756	$432
ETHERNET: 100Mbit/sec, PCI 32-bit	ETH101007-PCI		1	$120	6	$720	
ETHERNET: 10Mbit/sec, PCI 32-bit	ETH101-PCI		1	$130	12	$1,560	
MONITOR:14-inch Color	EVG142-COL		1	$219	6	$1,314	
KEYBD: 104 Key Spacesaver	PCK104-SKB		1	$34	6	$204	
MOUSE: 2 Button PS2	PWM1-PS2		1	$25	6	$150	
					Subtotal	**$31,704**	**$9,216**
Client Software							
Microsoft Windows NT Server 4.0, incl 5 CALs		Microsoft	3	$809	6	$4,854	$0
Microsoft SQL Workstation w/ Programmer's Toolkit		Microsoft	3	$499	1	$499	$0
Microsoft Visual C++ 32-bit edition (subscription)		Microsoft	3	$499	1	$499	$0
TUXEDO Core Functional Services 6.3 for NT		BEA	4	$3,000	6	$18,000	$13,500
					Subtotal	**$23,852**	**$13,500**
User Connectivity							
Ethernet Hub, 8-Port 100TX TrueFast + 10% spares	NX-H8TX	Netlux	5	$299	3	$897	spared
Ethernet Hub, 8-Port 10Base-T + 1-Port BNC + 10% spares	NX-H16EZ	Netlux	5	$83	736	$61,088	spared
					Subtotal	**$61,985**	**$0**
					Total	**$440,444**	**$56,383**

Notes:

1. HW Maintenance - 1st 36 months included in Unisys product costs. The next 24 months are at the level: Standard Performance-Gold.
2. All Microsoft maintenance is covered by the maintenance cost of Microsoft SQL Server.
3. 10% or minimum 2 spares are added in place of onsite service (products have a five year return-to-vendor warranty)
4. Pricing: 1 = Western Micro, 2= ALR, 3 = Microsoft, 4 = BEA, 5 = Netlux

Five Year Cost of Ownership	**$496,827**
TPC-C Throughput	**13,089.30**
$/tpmC	**$37.96**

The benchmark results and test methodology were audited by Richard Gimarc of Performance Metrics, Inc.

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumption about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmarks specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org.

NUMERICAL QUANTITIES SUMMARY

Unisys Aquanta HS/6 Server

Microsoft SQL Server 6.5 Enterprise Edition

MQTh, Computed Maximum Qualified Throughput: **13089.30**

% throughput difference, reported & reproducibility runs: 0.20%

Transaction Mix

New Order	44.75%
Payment	43.17%
Delivery	4.04%
Stock-Level	4.03%
Order-Status	4.01%

Response Times

Transaction	Average	Maximum	90th %ile
New-Order	0.54	5.24	0.78
Payment	0.56	5.20	0.71
Delivery	0.39	4.64	0.70
Stock-Level	2.31	7.81	3.80
Order Status	1.07	6.25	1.60
Menu	0.22	3.59	0.30
Delivery (Deferred)	3.42	16.82	9.73

Response time delay added for emulated components (seconds)

RT Response time	0.1
Menu Response time	0.1

Keying/Think Time Times (seconds)

Transaction	Minimum	Average	Maximum
New-Order	18.00/0	18.01/12.03	19.19/120.4
Payment	3.00/0	3/12.04	4.24/120.4
Delivery	2.00/0	2/5.08	3.15/50.7
Stock-Level	2.00/0	2/5.06	3.15/50.7
Order-Status	2.00/0	2/10.08	3.13/100.91

Test Duration

Ramp up time	93 minutes
Measurement interval (M)	30 minutes
Transactions (all types) completed during measurement interval	877586
Ramp-down time	23 minutes

Checkpointing:

Number of checkpoints	1
Checkpoint interval	30 minutes

Table of Contents

Abstract iv

Table of Contents viii

Preface xii

0. General Items 0-1

0.1. Order and Titles 0-1

0.2. Executive Summary Statement 0-1

0.3. Numerical Quantities Summary 0-1

0.4. Application Code Disclosure 0-1

0.5. Benchmark Sponsor 0-2

0.6. Parameter Settings 0-2

0.7. Configuration Diagrams 0-2

1. Clause 1: Logical Database Design 1-1

1.1. Table Definitions 1-1

1.2. Physical Organization of the Database 1-1

1.3. Insert and/or Delete Operations 1-1

1.4. Partitioning 1-1

1.5. Replication, Duplication or Additions 1-1

2. Clause 2: Transaction & Terminal Profiles 2-1

2.1. Random Number Generation 2-1

2.2. Input/Output Screen Layout 2-1

2.3. Priced Terminal Feature Verification 2-1

2.4. Presentation Managers or Intelligent Terminal 2-1

2.5. Transaction Statistics 2-1

2.6. Queuing Mechanism of Delivery 2-2

3. Clause 3: Transaction & System Properties 3-1

3.1. Transaction System Properties (ACID) 3-1

3.2. Atomicity 3-1

3.2.1. Completed Transaction 3-1

3.2.2. Aborted Transactions 3-1

3.3. Consistency 3-1

3.4. Isolation 3-2

3.5. Durability 3-2
3.5.1. Loss of Log and Loss of Data Disk 3-2
3.5.2. Instantaneous Interruption and Loss of Memory 3-3

4. Clause 4: Scaling & Database Population 4-1
4.1. Initial Cardinality of Tables 4-1
4.2. Constant Values 4-1
4.3. Database Layout 4-2
4.4. DBMS: Data Model and DBMS Interface/Access Language 4-2
4.5. DBMS Partitions/Replications 4-2
4.6. DBMS Space Requirements 4-2

5. Clause 5: Performance Metrics & Response Time 5-1
5.1. Measured Throughput (tpmC) 5-1
5.2. Response Times 5-1
5.3. Keying and Think Times 5-1
5.4. Response Time Frequency Distribution Curves 5-2
5.5. New Order Think Time Frequency Distribution Curve 5-5
5.6. Response Time versus Throughput Performance Curve 5-5
5.7. New-Order Throughput vs. Time 5-6
5.8. Determination of “Steady State” 5-6
5.9. Work Performed During Steady State 5-6
5.10. Reproducibility 5-7
5.11. Measurement Interval Duration 5-7
5.12. Regulation of Transaction Mix 5-7
5.13. Transaction Statistics 5-7
5.14. Checkpoint Statistics 5-8

6. Clause 6: SUT, Driver & Communications Definition 6-1
6.1. Remote Terminal Emulator (RTE) Description 6-1
6.2. Emulated Components 6-1
6.3. Functional Diagrams 6-1
6.4. Network Configuration 6-1
6.5. Network Bandwidth 6-1
6.6. Operator Intervention 6-2

7. Clause 7: Pricing 7-1
7.1. Pricing 7-1
7.1.1. System Pricing 7-1
7.1.2. Maintenance Pricing 7-1
7.1.3. Discounts 7-1
7.2. Availability 7-2
7.3. Measured tpmC, Price/Performance, and Availability Date 7-2

7.4. Country-Specific Pricing 7-2
7.5. Usage Pricing 7-2

8. Clause 8 : Full Disclosure Availability 8-1
8.1. Availability 8-1

9. Clause 9 : Audit 9-1
9.1. Auditor's Report 9-1

Appendix A - Client/Server Source A-1

Appendix B - Database Design B-1

Appendix C - Tunable Parameters C-1

Appendix D - RTE Code D-1

Appendix F - Third-Party Price Quotations F-1

Figures

Figure 0.1: Benchmarked Configuration 0-3
Figure 0.2: Priced Configuration 0-3
Figure 5.1: New Order Response Time Distribution 5-2
Figure 5.2: Payment Response Time Distribution 5-3
Figure 5.3: Order Status Response Time Distribution 5-3
Figure 5.4: Delivery Response Time Distribution 5-4
Figure 5.5: Stock Level Response Time Distribution 5-4
Figure 5.6: New Order Think Time Distribution 5-5
Figure 5.7: Response Time versus Throughput 5-5
Figure 5.8: Throughput (tpmC) versus Time 5-6

Tables

Table 2.1: Transaction Statistics 2-2
Table 4.1: Initial Cardinality of Database Table 4-1
Table 4.2: Constant C for NURand 4-1
Table 4.3: Disk Cage Configuration 4-3
Table 4.4: Disk Usage/Size Totals 4-4
Table 4.5: Disk Administrator Configuration 4-5
Table 5.1: Response Time Data 5-1
Table 5.2: Keying Times 5-1
Table 5.3: Think Times 5-2
Table 5.4: Transaction Statistics 5-8

Document Structure

The TPC Benchmark C Standard Specification requires test sponsors to publish, submit to the TPC, and make available to the public, a full disclosure report for any result to be considered compliant with the specification. The required contents of the full disclosure report are specified in Clause 8.

This report is submitted to satisfy the specification's requirement for full disclosure. It documents the compliance of the benchmark implementation and execution reported for the Unisys Corporation Aquanta HS/6 Server using Microsoft Windows NT 4.0 Enterprise Edition and Microsoft SQL Server 6.5 Enterprise Edition.

TPC Benchmark C Overview

The TPC Benchmark™ C Standard Specification Revision 3.3.2 was developed by the Transaction Processing Council (TPC). It is the intent of the TPC to develop a suite of benchmarks to measure the performance of computer systems executing a wide range of applications. Unisys and Microsoft Corporations are active participants in the TPC to define and develop such a suite of benchmarks.

TPC Benchmark™ C (TPC-C) is an OLTP workload. It is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

- The simultaneous execution of multiple transaction types that span a breadth of complexity.
- On-line and deferred transaction execution modes.
- Multiple on-line terminal sessions.
- Moderate system and application execution time.
- Significant disk input/output.
- Transaction integrity (ACID properties).
- Non-uniform distribution of data access through primary and secondary keys.
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships.
- Contention on data access and update.

The performance metric reported by TPC-C is a "business throughput" measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC). To be compliant with the TPC-C standard, all references to tpmC results must include the tpmC rate, the associated price-per-tpmC, and the availability date of the priced configuration.

Despite the fact that this benchmark offers a rich environment that emulates many OLTP environments, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

0. General Items

0.1. Order and Titles

The order and titles of sections in the Test Sponsor's Full Disclosure report must correspond with the order and titles of sections from the TPC-C standard specification (i.e., this document). The intent is to make it as easy as possible for readers to compare and contrast material in different Full Disclosure reports.

The order and titles of the sections in this report correspond with those from the TPC-C standard specification.

0.2. Executive Summary Statement

The TPC Executive Summary Statement must be included near the beginning of the Full Disclosure report.

The TPC Executive Summary Statement is included near the beginning of this report.

0.3. Numerical Quantities Summary

The numerical quantities listed below must be summarized near the beginning of the Full Disclosure report :

- *measurement interval in minutes,*
- *number of checkpoints in the measurement interval,*
- *checkpoint interval in minutes,*
- *number of transactions (all types) completed within the measurement interval,*
- *computed Maximum Qualified Throughput in tpmC,*
- *percentage difference between reported throughput and throughput obtained in reproducibility run,*
- *ninetieth percentile, average and maximum response times for the New-Order, Payment, Order-Status, Stock-Level, Delivery (deferred and interactive) and Menu transactions,*
- *time in seconds added to response time to compensate for delays associated with emulated components,*
- *percentage of transaction mix for each transaction type.*

These numerical quantities are summarized near the beginning of this report.

0.4. Application Code Disclosure

The applicable program (as defined in Clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input and output functions.

Appendix A contains the client application code used in this TPC-C benchmark. Appendix B contains the SQL stored procedures which implement the TPC-C transactions.

0.5. Benchmark Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

This TPC benchmark C was sponsored by Unisys Corporation. The benchmark test was developed by Microsoft and Unisys. The benchmark was conducted at Unisys, Mission Viejo, California.

0.6. Parameter Settings

Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including but not limited to:

- *Data Base tuning options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters*

Appendix C contains the configuration and system parameters used in running these TPC-C tests. It also contains all the client and server OS, and SQL Server tunable parameters.

0.7. Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

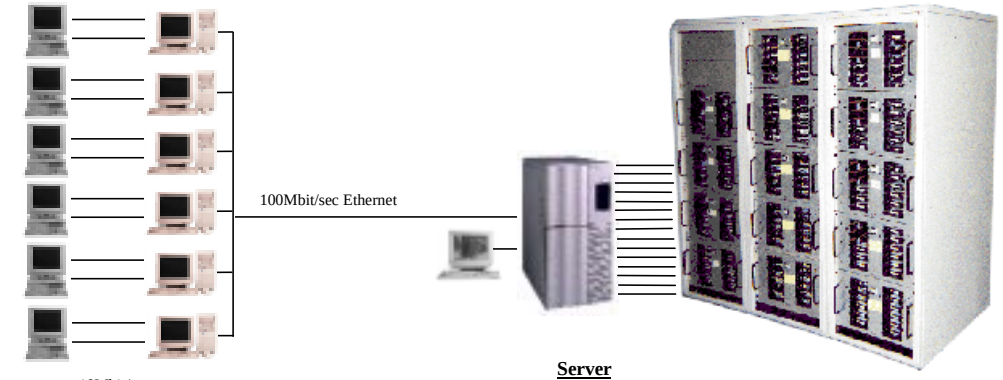
- *Number and type of processors.*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test.*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including their protocol type.*
- *Number of LAN (e.g., Ethernet) connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure (see Clause 8.1.8).*
- *Type and the run-time execution location of software components (e.g., DBMS, client processes, transactio n monitors, software drivers, etc.).*

The Remote Terminal Emulator (RTE) software used for these TPC-C tests is proprietary to Unisys. The benchmarked configuration of the RTE and Aquanta HS/6 server is illustrated in Figure 0.1. Tables 4.3, 4.4 and 4.5 contain a detailed explanation of the disk configuration.

The priced configuration for the Aquanta HS/6 server is shown in Figure 0.2.

Figure 0.1: Benchmarked Configuration

Aquanta HS/6 Server - Benchmarked Configuration



6 RTEs
emulating
10,650 users

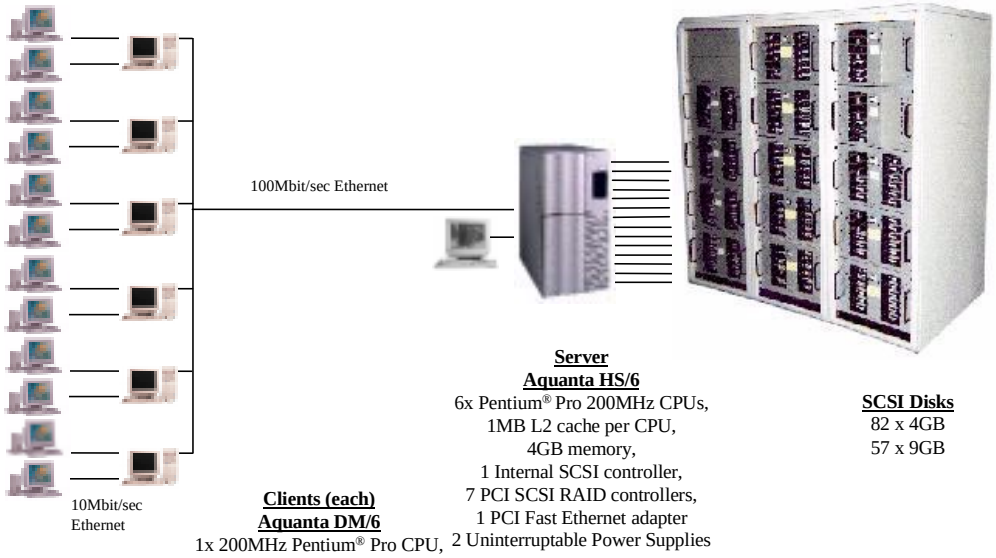
Clients (each)
Aquanta DM/6 II
1x 200MHz Pentium® Pro CPU,
256KB L2 cache
256MB memory,
1 x 1.5GB disk,
1 PCI Fast Ethernet adapter for Server connection,
2 PCI Ethernet adapters for RTE connections

Server
Aquanta HS/6
6x Pentium® Pro 200MHz CPUs,
1MB L2 cache per CPU,
4GB memory,
1 Internal SCSI controller,
7 PCI SCSI RAID controllers,
1 PCI Fast Ethernet adapter

SCSI Disks
82 x 4GB
48 x 9GB
12 x 9GB (backup)

Figure 0.2: Priced Configuration

Aquanta HS/6 Server - Priced Configuration



10,650 NetPC's

Clients (each)
Aquanta DM/6
1x 200MHz Pentium® Pro CPU,
256KB L2 cache
256MB memory,
1 x 1.5GB disk,
1 PCI Fast Ethernet adapter for Server connection,
2 PCI Ethernet adapters for RTE connections

Server
Aquanta HS/6
6x Pentium® Pro 200MHz CPUs,
1MB L2 cache per CPU,
4GB memory,
1 Internal SCSI controller,
7 PCI SCSI RAID controllers,
1 PCI Fast Ethernet adapter
2 Uninterruptable Power Supplies

SCSI Disks
82 x 4GB
57 x 9GB

1. Clause 1: Logical Database Design

1.1. Table Definitions

Listings must be provided for all table definition statements and all other statements used to setup the data base.

Appendix B contains the SQL definitions of all the required database devices, tables, indexes and stored procedures, plus a listing of the program used to load the database and establish the required initial populations of each table.

1.2. Physical Organization of the Database

The physical organization of tables and indices, within the data base, must be disclosed.

The disk space was allocated to SQL Server according to the data in Table 4.4. The SQL definitions are contained in Appendix B.

1.3. Insert and/or Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restriction in the SUT data base implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.

There were no restrictions on insert and/or delete operations to any of the tables.

1.4. Partitioning

While there are few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark, any such partitioning must be disclosed.

Partitioning was not used for any table in this implementation.

1.5. Replication, Duplication or Additions

Replication of tables, if used, must be disclosed.

Additional and/or duplicate attributes in any table must be disclosed along with a statement on the impact on performance.

No replications, duplications or additional attributes were used in this implementation.

2. Clause 2: Transaction & Terminal Profiles

2.1. Random Number Generation

The method of verification for the random number generation must be disclosed.

The drivers used the Unisys RTE program, which was independently audited. The initial population of the database was performed by the loader program from V3-02 of the Microsoft TPC-C toolkit, which was also independently audited. Furthermore, the auditor sampled various initial and runtime distributions produced by this implementation to verify correctness.

2.2. Input/Output Screen Layout

The actual layout of the terminal input/output screens must be disclosed.

The screen layouts are based on those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3. and 2.8.3 of the TPC Benchmark C Standard Specification. There are some minor differences in appearance due to the use of a web client implementation.

2.3. Priced Terminal Feature Verification

The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).

This was verified by the auditor by a direct experiment during the onsite audit portion of this benchmark, using Microsoft Internet Explorer 3.0 as the web browser.

2.4. Presentation Managers or Intelligent Terminal

Any usage of presentation managers or intelligent terminals must be explained.

Application code running on the client implemented the TPC-C user interface. A listing of this code is included in Appendix A. No presentation manager was used on the client, as screen manipulation and data input/output was handled for each user by the Microsoft Internet Explorer web browser running on each user PC.

2.5. Transaction Statistics

The percentage of New-Order transactions that were rolled back as a result of an unused item number must be disclosed.

The number of items per order entered by New-Order transactions must be disclosed.

The percentage of home and remote Payment transactions must be disclosed.

The percentage of Payment and Order-Status transactions that used non-primary key (C_LAST) access to the database must be disclosed.

The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW-ORDER table must be disclosed.

The mix (i.e., percentages) of transaction types seen by the SUT must be disclosed.

Table 2.1 contains all these statistics.

Table 2.1: Transaction Statistics

Transaction Type	Statistics	Value
New Order	Rolledback transactions	1.01%
	Home warehouse	98.99%
	Remote warehouse	1.01%
	Average Items per Order	9.99
Payment	Home warehouse	84.96%
	Remote warehouse	15.04%
	Non-primary key access	60.03%
Order Status	Non-primary key access	60.10%
Delivery	Skipped transactions (Interactive)	0
	Skipped transaction counts (Deferred)	0
	Skipped District counts (Deferred)	0
Transaction Mix	New Order	44.75%
	Payment	43.17%
	Delivery	4.04%
	Stock-Level	4.03%
	Order-Status	4.01%

2.6. Queuing Mechanism of Delivery

The queuing mechanism used to defer execution of the Delivery transaction must be disclosed.

Tuxedo provides the queue for delivery servers. The client application process posts delivery transactions to the delivery queue using a Tuxedo asynchronous call with the TPNoReply option. Upon return from this call, the client application provides a 'delivery queued' response to the user. Delivery servers independently retrieve messages from their queue, submit them to the data base for processing, and log the result to a file upon completion. The source code for this delivery process is included in Appendix A.

3. Clause 3: Transaction & System Properties

3.1. Transaction System Properties (ACID)

The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7.

The TPC Benchmark C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation, and Durability (ACID).

This section defines each of these properties, describes the steps taken to ensure that they were present during the test and describes a series of tests done to demonstrate compliance with the specification. All ACID property tests were executed successfully.

3.2. Atomicity

The system under test must guarantee that data base transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data.

3.2.1. Completed Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately.

The balances from a randomly selected warehouse, district, and customer row were retrieved by customer number from a script. A Payment transaction was submitted with the same warehouse, district and customer identifiers for a known amount. After completion of the Payment transaction, the balances of the selected warehouse, district, and customer were again retrieved to verify that the changes had been made correctly.

3.2.2. Aborted Transactions

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have NOT been changed.

The balances from a randomly selected warehouse, district, and customer row were retrieved by customer number from a script. A Payment transaction was submitted with the same warehouse, district and customer identifiers that issued a ROLLBACK command rather than a COMMIT. After the transaction completed, the balances of the selected warehouse, district, and customer were again retrieved to verify that no changes had been made to the database.

3.3. Consistency

Consistency is the property of the application that requires any execution of a data base transaction to take the data base from one consistent state to another, assuming that the data base is initially in a consistent state.

The benchmark specification requires explicit demonstration of the following four consistency conditions:

1. The sum of the district balances in a warehouse is equal to the warehouse balance;
2. For each district, the next order id minus one is equal to maximum order id in the ORDER table and equal to the maximum new order id in the NEW ORDER table;
3. For each district, the maximum order id minus minimum order id in the ORDER table plus one equals the number of rows in the NEW-ORDER table for that district;
4. For each district, the sum of the order line counts in the ORDER table equals the number of rows in the ORDER-LINE table for that district;

In order to demonstrate this consistency, the following steps were taken:

1. Prior to the start of a benchmark run, the consistency of the database was verified by testing successfully conditions 1-4 described above with a script.
2. A run under full user load was executed for over 10 minutes with a checkpoint during the run.
3. After completion of that test, the consistency of the database was again verified by successfully testing using the same consistency script as in step 1.

3.4. Isolation

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation defined above (clause 3.4.1) is obtained.

The benchmark specification defines seven required tests to be performed to demonstrate that required levels of transaction isolation are met. These tests, described in Clauses 3.4.2.1 - 3.4.2.7, were all performed from a script and verified by the auditor. In Isolation Test 7, Case A was observed. In addition, the phantom tests and stock level tests were executed and verified to be successful.

3.5. Durability

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure data base consistency after recovery from any one of the failures listed in Clause 3.5.3.

Three durability tests were executed to satisfy the requirements of the specification. The test for loss of memory and instantaneous interruption was combined and performed with a fully scaled database with 10,650 emulated users. The loss of log and loss of data tests were combined and performed on a separate ten warehouse database with 100 emulated users. To the best of our knowledge, these tests prove that the fully scaled configuration used for the throughput test would also meet all durability tests.

3.5.1. Loss of Log and Loss of Data Disk

The following steps were taken (using a ten warehouse database on the SUT) to demonstrate durability in the case of loss of a log and subsequent loss of a data disk:

1. The database was backed up to extra disks on a dump device.
2. The D_NEXT_O_ID fields for all rows in the district table were summed up to determine the initial count of orders present in the database.
3. The RTE was started with 100 users. On the driver systems, committed and rolled back New-Order transactions were recorded in a “success” file.
4. One of the mirrored hot-pluggable log disks was removed from the disk cabinet after five minutes of steady state.

5. Log disk mirroring is done by a RAID controller, so NT and SQL Server did not notice the disk loss. The benchmark run continued without interruption.
6. After another five minutes of running at steady state, a hot-pluggable database disk was removed from the disk cabinet.
7. NT and SQL Server encountered IO errors due to the missing disk and recorded these errors in the NT event log and SQL Server error log, respectively.
8. First, the RTEs and clients were stopped, then SQL Server was stopped, and finally the SUT was shutdown and restarted.
9. SQL Server was restarted and marked the database as 'suspect'. A dump of the transaction log was taken to extra disks on a dump device.
10. Next, scripts were executed to drop the database and all its devices. Then, SQL Server was shutdown again and the SUT shutdown.
11. A different data disk was inserted in the disk cabinet to replace the one removed. (The removed log disk was never replaced.) The RAID controller was used to recreate the stripe set containing the new data disk.
12. The SUT was restarted, and Disk Administrator was used to assign the proper drive letter to the new stripe set. SQL Server was then restarted and an empty database created. After space allocation had finished, the database was recovered by first reloading the entire initial database and log from backup, then by loading and applying the transaction log dump that was taken after the data disk failure. The latter step restored all committed transactions to the database.
13. Consistency condition 3 of Clause 3.3.2.3 was executed to verify database consistency.
14. Step 2 was repeated to determine the total number of orders. This number was subtracted from the count obtained previously in Step 2 to determine the number of additional orders added to the database.
15. The contents of the "success" files on the drivers were sampled to verify that the records in the "success" file for committed New-Order transactions had corresponding records in the ORDER table and no entries existed for rolled back transactions. Moreover, the counts were matched with those obtained in step 14.

3.5.2. Instantaneous Interruption and Loss of Memory

Instantaneous interruption and loss of memory tests were combined because the loss of power erased the contents of memory. This failure was induced by removing the primary power to the System Under Test while the benchmark was executing.

1. The D_NEXT_O_ID fields for all rows in the district table were summed up to determine the initial count of orders present in the database (count1).
2. On the driver systems, committed and rolled back New-Order transaction were recorded in a "success" file.
3. The benchmark was executed at full load with 10,650 emulated users for a minimum of 10 minutes.
4. Immediately after execution of a checkpoint completed, the system's primary power was turned off.
5. The test was aborted on the driver and client systems, and the RTEs and clients were shutdown.
6. Power was restored to the SUT, the system rebooted, SQL Server was restarted, and automatic database recovery was performed. The database recovery uses the transaction log to reapply all committed transactions and rollback any (in progress) uncommitted transactions, so that the database disks are correct.
7. After recovery finished, Consistency Condition of Clause 3.3.2.3 (no gaps in NO_O_ID) was executed to verify that the database was consistent..
8. Next, samples of the contents of the "success" file on the driver were compared against corresponding rows of the ORDER table to verify that records in the "success" file for committed New-Order transactions had corresponding records in the ORDER table and no entries existed for rolled back transactions.
9. Finally, step 1 was repeated to determine the total number of orders (count2). Count2 minus count1 was not less than the number of committed New-Order records in the "success" file.

4. Clause 4: Scaling & Database Population

4.1. Initial Cardinality of Tables

The Cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run, must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2 and the Auditor's attestation letter) the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

The TPC-C database for this test was configured with 1,110 warehouses. The cardinality of each table in the database is listed in Table 4.1

Table 4.1: Initial Cardinality of Database Table

Table	Occurrences
Warehouse	1,110
District	11,100
Customer	33,300,000
History	33,300,000
Order	33,300,000
New Order	9,990,000
Order Line	332,999,230
Stock	111,000,000
Item	100,000

45 rows were deleted from the warehouse table before executing the measurement runs.

4.2. Constant Values

The following values were used as the constant C input values to the NURand function during Build and Run time for this implementation.

Table 4.2: Constant C for NURand

Function	Value
C_LAST (Build)	123
C_LAST (Run)	223

4.3. Database Layout

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems.

Tables 4.3, 4.4 and 4.5 list the distribution of the database over 121 disks and the transaction log over four mirrored pairs of disks for the benchmark configuration. In addition, there was one disk containing Windows NT Enterprise Edition and SQL Server Enterprise Edition code and the Master database plus the paging file. Database backup used an extra 12 disks. For Durability testing with a smaller 10 warehouse database, another 8 disks were used: 4 for the database, 2 for a mirrored log and 2 for backup. All these 20 extra disks were excluded from the priced configuration. Nine extra database disks were added to the priced configuration to meet the 180-day spare required.

4.4. DBMS: Data Model and DBMS Interface/Access Language

A statement must be provided that describes:

1. *The data model implemented by the DBMS used (e.g., relational, network, hierarchical).*

2. *The database interface (e.g., embedded, call level) and access language (e.g., SQL, DL/1, COBOL, read/write) used to implement the TPC-C transactions. If more than one interface/access language is used to implement TPC-C, each interface/access language must be described and a list of which interface/access language is used with which transaction type must be disclosed.*

Microsoft SQL Server 6.5 Enterprise Edition is a relational DBMS.

The client software interfaced to SQL Server through Stored Procedures invoked through Remote Procedure Calls embedded in the C application code. Specifically, DBLIB and TCP/IP sockets were used.

4.5. DBMS Partitions/Replications

The mapping of database partitions/replications must be explicitly described.

No table partitioning or replication was done.

4.6. DBMS Space Requirements

Details of the 180 day space computation along with proof that the database is configured to sustain 8 hours of growth for dynamic tables (Order, Order-line, and History) must be disclosed (see Clause 4.2.3).

Appendix E lists the space requirements for the 180-day space as well as the logical log space for eight hours.

Table 4.3: Disk Cage Configuration

Disk Cage Configuration for TPCC

Adapter 1

Channel 1				Channel 2			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
13	4GB	empty	6	13	empty	empty	6
12	4GB	4GB	4	12	4GB	4GB	4
11	4GB	4GB	3	11	4GB	4GB	3
10	4GB	4GB	2	10	4GB	4GB	2
9	4GB	4GB	1	9	4GB	4GB	1
8	4GB	4GB	0	8	4GB	4GB	0

Adapter 2

Channel 1				Channel 2			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
13	empty	empty	6	13	empty	empty	6
12	4GB	4GB	4	12	4GB	4GB	4
11	4GB	4GB	3	11	4GB	4GB	3
10	4GB	4GB	2	10	4GB	4GB	2
9	4GB	4GB	1	9	4GB	4GB	1
8	4GB	4GB	0	8	4GB	4GB	0

Adapter 3

Channel 1				Channel 2			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
13	empty	empty	6	13	empty	empty	6
12	4GB	9GB	4	12	4GB	9GB	4
11	4GB	9GB	3	11	4GB	9GB	3
10	4GB	9GB	2	10	4GB	9GB	2
9	4GB	9GB	1	9	4GB	9GB	1
8	4GB	9GB	0	8	4GB	9GB	0

Adapter 4

Channel 1				Channel 2			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
13	empty	empty	6	13	empty	empty	6
12	4GB	9GB	4	12	4GB	9GB	4
11	4GB	9GB	3	11	4GB	9GB	3
10	4GB	9GB	2	10	4GB	9GB	2
9	4GB	9GB	1	9	4GB	9GB	1
8	4GB	9GB	0	8	4GB	9GB	0

Adapter 5

Channel 1				Channel 2			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
13	empty	empty	6	13	empty	empty	6
12	4GB	9GB	4	12	4GB	9GB	4
11	4GB	9GB	3	11	4GB	9GB	3
10	4GB	9GB	2	10	4GB	9GB	2
9	4GB	9GB	1	9	4GB	9GB	1
8	4GB	9GB	0	8	4GB	9GB	0

Adapter 6

Channel 1				Channel 2			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
13	empty	empty	6	13	empty	empty	6
12	4GB	9GB	4	12	4GB	9GB	4
11	4GB	9GB	3	11	4GB	9GB	3
10	4GB	9GB	2	10	4GB	9GB	2
9	4GB	9GB	1	9	4GB	9GB	1
8	4GB	9GB	0	8	4GB	9GB	0

Adapter 7

Channel 2		Channel 1		Channel 3			
ID	Left Side	Right Side	ID	ID	Left Side	Right Side	ID
6	empty	empty	6	13	9GB	9GB	6
4	9GB	9GB	4	12	9GB	9GB	4
3	9GB	9GB	3	11	9GB	9GB	3
2	9GB	9GB	2	10	9GB	9GB	2
1	9GB	9GB	1	9	9GB	9GB	1
0	empty	empty	0	8	9GB	9GB	0

Table 4.4: Disk Usage/Size Totals

Disk Usage/Size Totals for TPCC

Disk Usage	Size (GB)	Adaptec	HA-1	HA-2	HA-3	HA-4	HA-5	HA-6	HA-7	Extra	Total
system	4	1									1
tpcc - data	4		21	20	10	10	10	10			81
tpcc - data	9				10	10	10	10			40
tpcc - log	9								8		8
tpcc - backup	9								12		12
Total Measured	4 & 9	1	21	20	20	20	20	20	20		142
4GB drives	4	1	21	20	10	10	10	10			82
9GB drives	9				10	10	10	10	20		60
Total Priced	4 & 9	1	21	20	20	20	20	20	8	9	139
4GB drives	4	1	21	20	10	10	10	10			82
9GB drives	9				10	10	10	10	8	9	57

Table 4.5: Disk Administrator Configuration

Disk Administrator Configuration for HS/6

Disk	Partition 1	Partition 2	Partition 3	Unused Capacity	HA#	LD#	Usage
0	unassigned	J:	P:		1	1	
87126MB	(raw) 1004 MB	(raw) 10503 MB	(raw) 5005 MB	Free Space 70614 MB			data tpcc
1	E:	K:	unassigned		2	1	
82976MB	(raw) 1004 MB	(raw) 13500 MB	(raw) 5005 MB	Free Space 63468 MB			data tpcc
2	H:	L:	R:		3	1	
128308 MB	(raw) 1004 MB	(raw) 10503 MB	(raw) 5005 MB	Free Space 111796 MB			data tpcc
3	I:	M:	S:		4	1	
128308 MB	(raw) 1004 MB	(raw) 10503 MB	(raw) 5005 MB	Free Space 111796 MB			data tpcc
4	F:	N:	T:		5	1	
128308 MB	(raw) 1004 MB	(raw) 10503 MB	(raw) 5005 MB	Free Space 111796 MB			data tpcc
5	G:	O:	U:		6	1	
128308 MB	(raw) 1004 MB	(raw) 10503 MB	(raw) 5005 MB	Free Space 111796 MB			data tpcc
6	V:				7	1	
34726 MB	(raw) 34726 MB			(none)			log tpcc
7	W: BACKUP				7	2	
86812 MB	NTFS 86812 MB			(none)			backup tpcc
8	C: SYSTEM	X: FILES			Adaptec		
4149 MB	NTFS 2047 MB	NTFS 2102 MB		(none)			system files
CD-ROM	D:				Adaptec		

5. Clause 5: Performance Metrics & Response Time

5.1. Measured Throughput (tpmC)

Measured tpmC must be reported.

The measured tpmC was 13,089.30.

5.2. Response Times

Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the Menu response time.

Table 5.1: Response Time Data

Transaction	Average	Maximum	90th %ile
New-Order	0.54	5.24	0.78
Payment	0.56	5.20	0.71
Delivery	0.39	4.64	0.70
Stock-Level	2.31	7.81	3.80
Order Status	1.07	6.25	1.60
Menu	0.22	3.59	0.30
Delivery (Deferred)	3.42	16.82	9.73

5.3. Keying and Think Times

The minimum, the average, and the maximum keying and think times must be reported for each transaction type.

Table 5.2: Keying Times

Transaction	Minimum	Average	Maximum
New-Order	18.00	18.01	19.19
Payment	3.00	3.00	4.24
Delivery	2.00	2.00	3.15
Stock-Level	2.00	2.00	3.15
Order Status	2.00	2.00	3.13

Table 5.3: Think Times

Transaction	Minimum	Average	Maximum
New-Order	0.00	12.03	120.40
Payment	0.00	12.04	120.40
Delivery	0.00	5.08	50.70
Stock-Level	0.00	5.06	50.70
Order Status	0.00	10.08	100.91

5.4. Response Time Frequency Distribution Curves

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type.

Figure 5.1: New Order Response Time Distribution

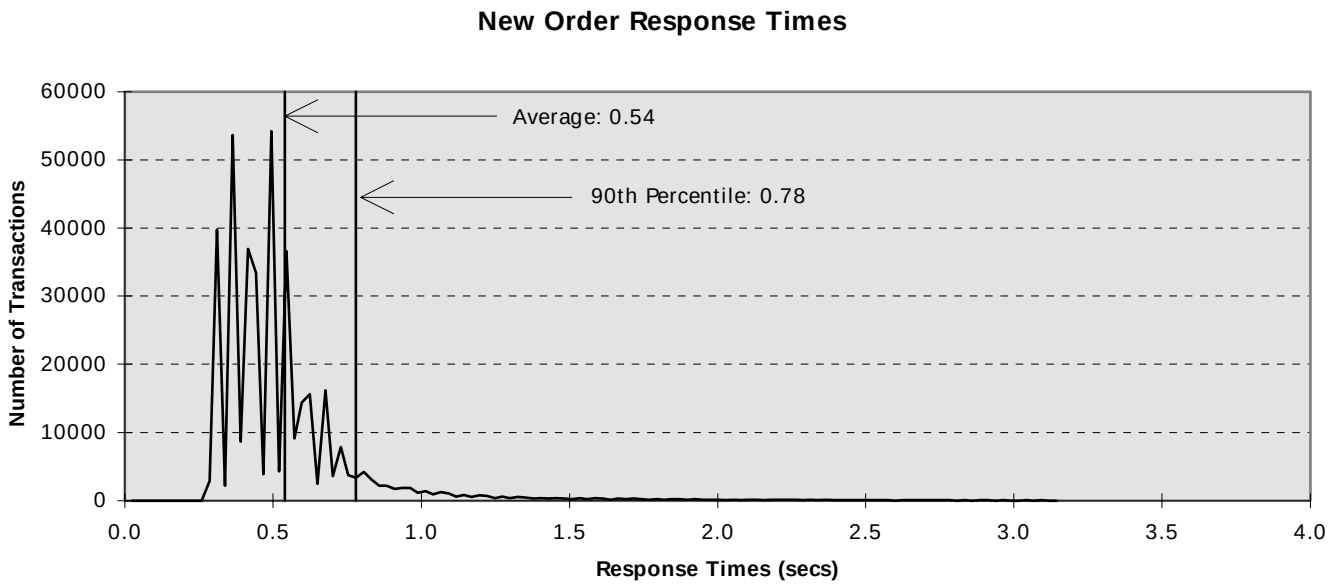


Figure 5.2: Payment Response Time Distribution

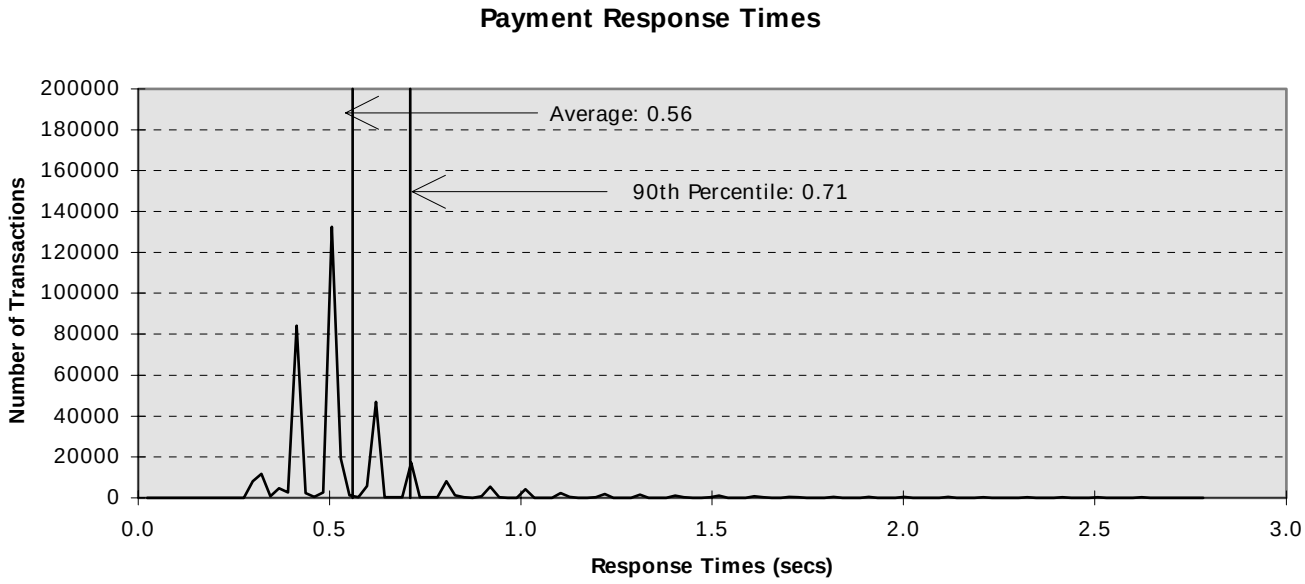


Figure 5.3: Order Status Response Time Distribution

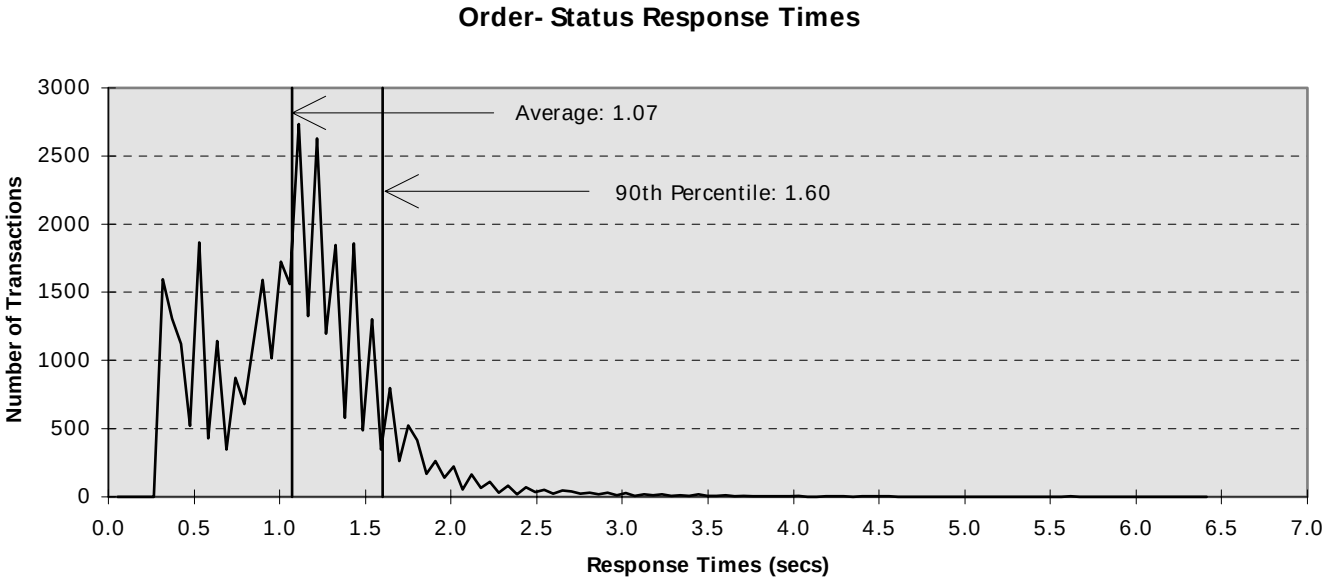


Figure 5.4: Delivery Response Time Distribution

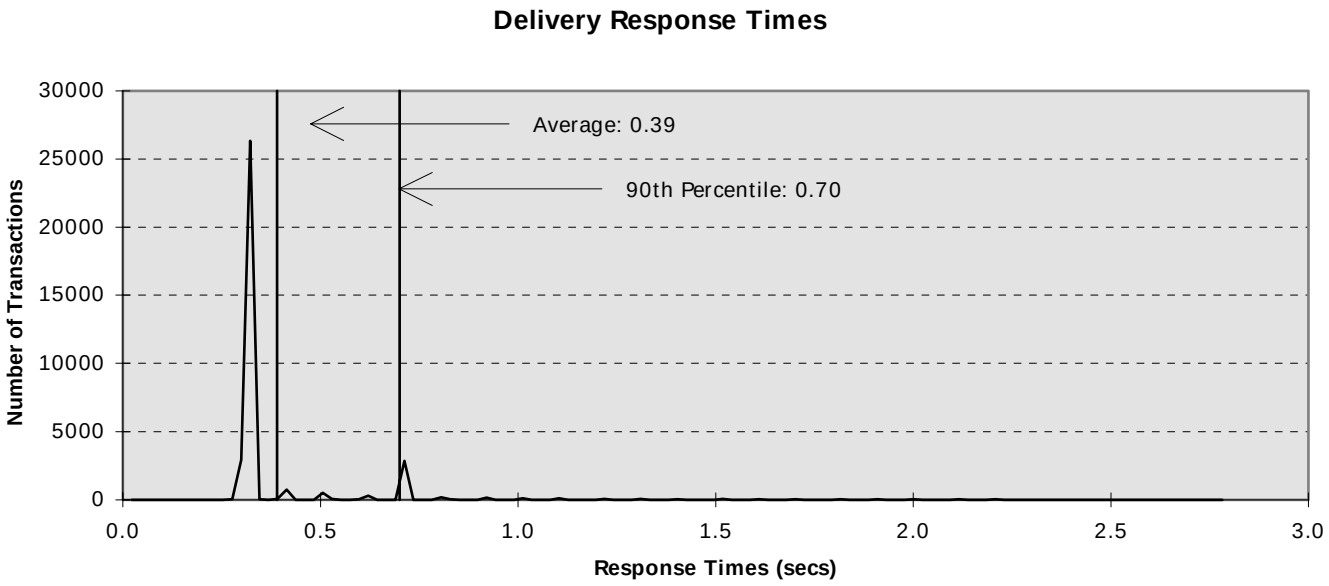
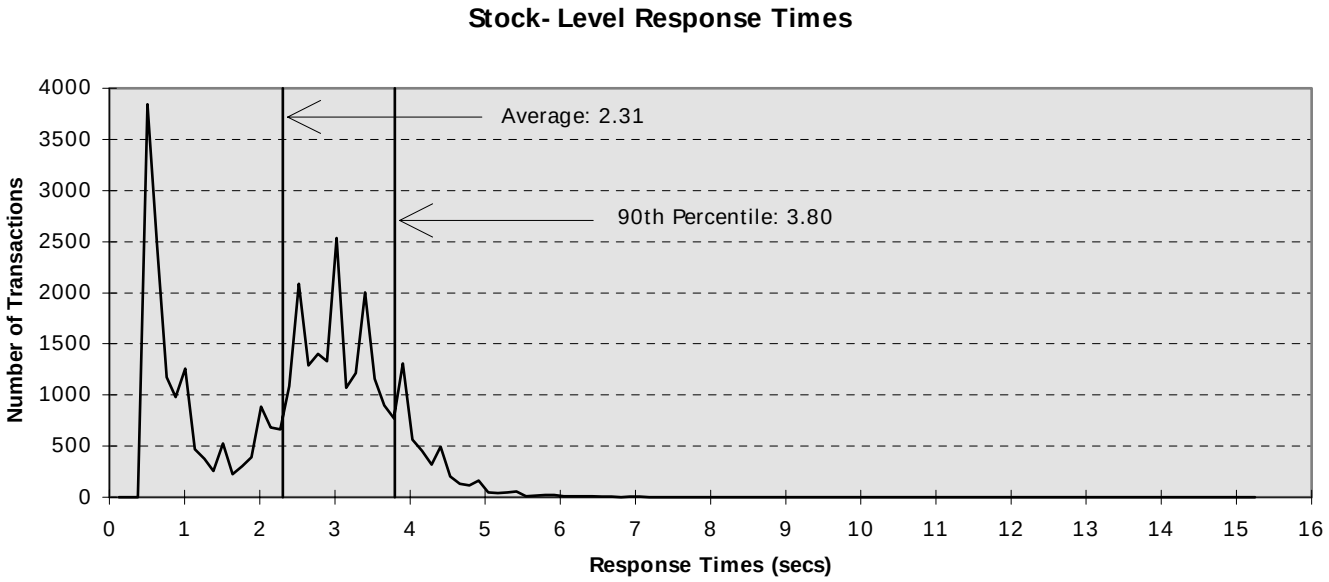


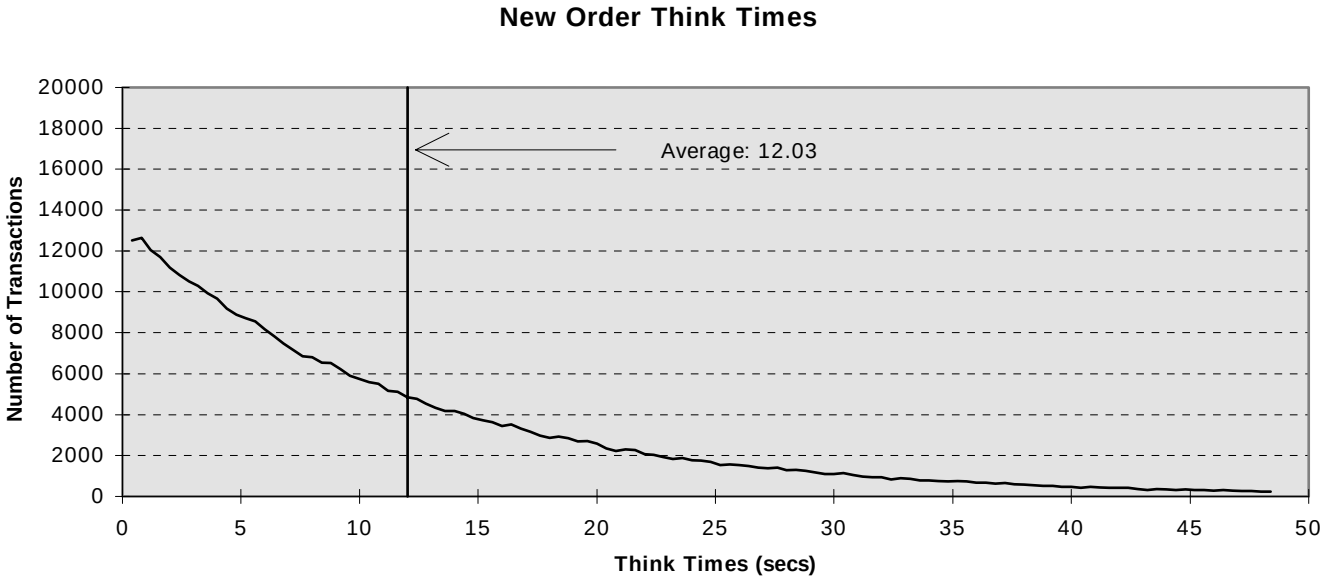
Figure 5.5: Stock Level Response Time Distribution



5.5. New Order Think Time Frequency Distribution Curve

Think Time frequency distribution curve (see Clause 5.6.3) must be reported for the New-Order transaction.

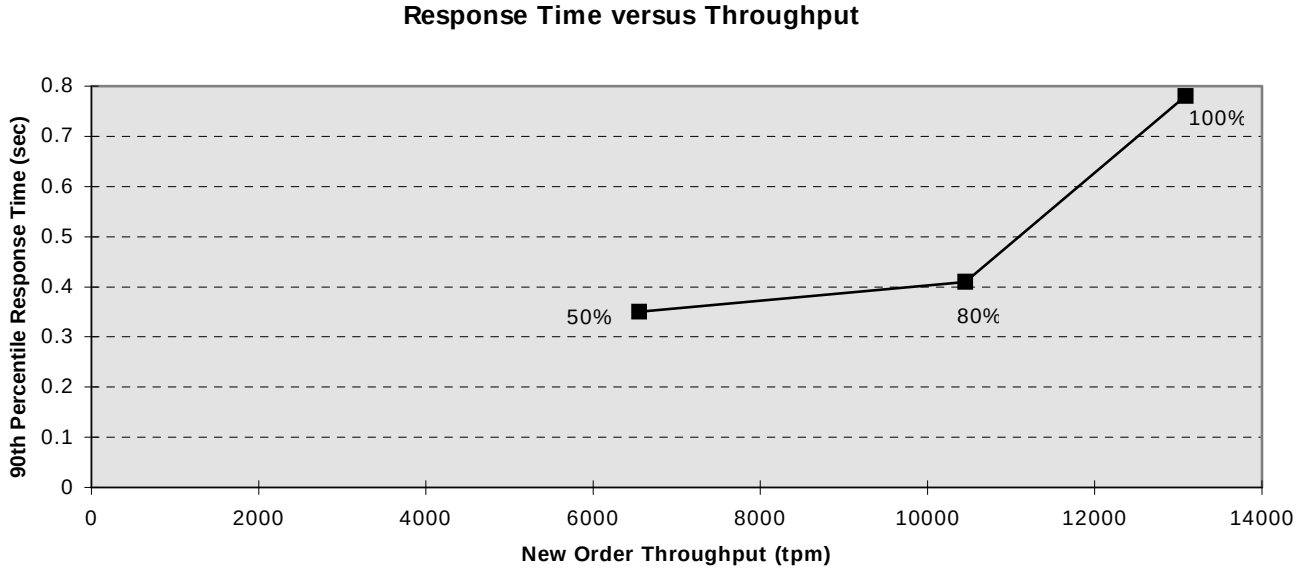
Figure 5.6: New Order Think Time Distribution



5.6. Response Time versus Throughput Performance Curve

The performance curve for response times versus throughput (Clause 5.6.2) must be reported for the New-Order transaction

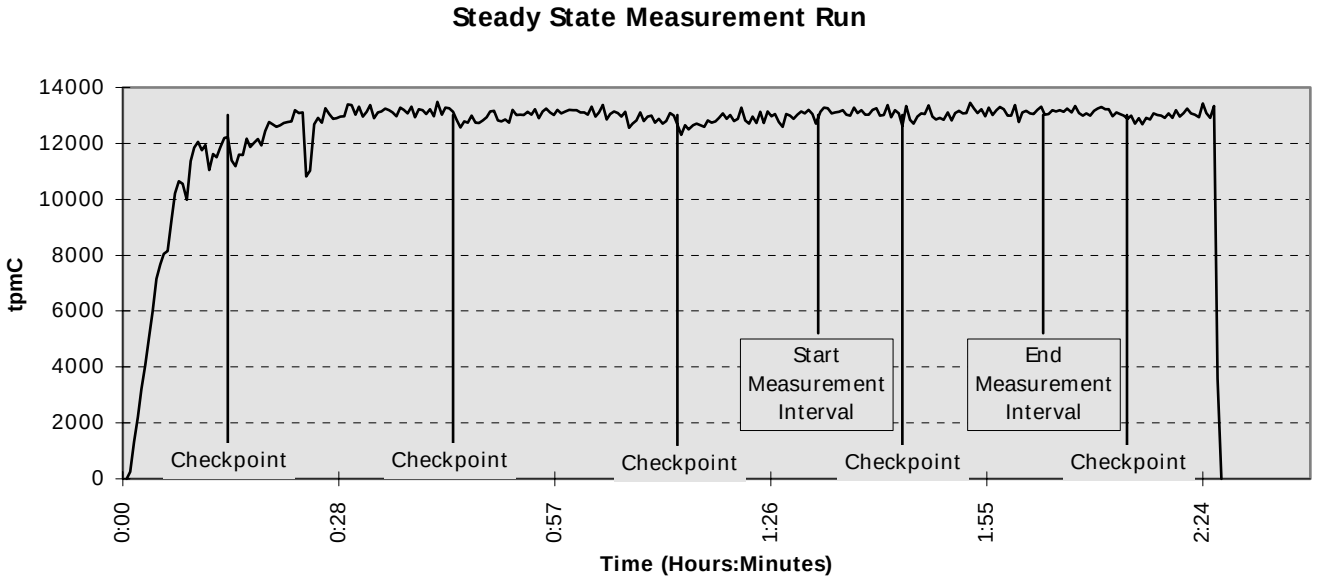
Figure 5.7: Response Time versus Throughput



5.7. New-Order Throughput vs. Time

A graph of throughput versus elapsed time (Clause 5.6.5) must be reported for the New-Order transaction.

Figure 5.8: Throughput (tpmC) versus Time



5.8. Determination of "Steady State"

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval (see Clause 5.5) must be described.

The transaction throughput rate (tpmC) and response time were relatively constant after the initial 'ramp up' period. The throughput and response time behavior were determined by examining data reported for each 30-second interval over the duration of the benchmark. Ramp-up, steady state, and ramp-down regions are discernible in the graph presented in Figure 5.8.

5.9. Work Performed During Steady State

A description of how the work normally performed during a sustained test (for example checkpointing, writing redo/undo log records, etc.) actually occurred during the measurement interval must be reported.

The RTE selects a transaction type from the menu and prepares to request the appropriate blank form. A timestamp is taken before the form request is sent and after the response is returned. The difference between the two is saved off as the menu response time. The RTE then generates input data for the transaction to create a completed form and waits the appropriate key time. A timestamp is taken before the completed form is sent and after the response is returned. The difference between these two is saved off as the transaction response time. Both response times are padded with a 0.1 second delay per spec to account for the web browser delay. The appropriate transaction data and response times are logged and the RTE waits the required think time interval before repeating the process. Each RTE driver maintains its own log file. Log file contents are consolidated for the reports.

The RTE emulates web browsers (not terminals) in this client server implementation. The RTE sends and receives HTML formatted data using HTTP through Ethernet LANs to a client application running under Microsoft's IIS web

server on the client machine. The client application processes the request, sends the transaction to a Tuxedo TPC-C application server queue, waits for the transaction response (except for delivery), and returns an appropriately formatted HTML form back to the (emulated) web browser (RTE). The Tuxedo TPC-C application server retrieves a message from its queue, invokes request processing via a stored procedure on the database server using Microsoft SQL Server DBLIB and RPC through sockets over another Ethernet LAN, accepts the response, and returns a result to the client application (via Tuxedo). For delivery transactions, the client application does not wait for the Tuxedo TPC-C delivery server to respond. Each delivery server logs its results to its own file. The delivery report files are consolidated for reports.

To perform checkpoints at specific intervals, SQL Server's checkpoint interval was set to the maximum allowable value and a script was written to schedule checkpoints at 30 minute intervals and record the start and end time of each checkpoint. The checkpoint script was started manually on one of the client machines after the RTE had all users logged in and sending transactions and a steady state had been achieved. Using this information, the positioning of the checkpoint within the measurement interval was verified to be clear of the guard zones.

At each checkpoint, SQL Server wrote to disk all database pages in memory that had been updated but not yet physically written to the disk. Upon completion of the checkpoint, SQL Server also wrote records to the transaction log indicating that a checkpoint had completed.

5.10. Reproducibility

A description of the method used to determine the reproducibility of the measurement results must be reported.

In a repeat test, carried out in the same manner as the primary test, a throughput of 13,063.47 tpmC was achieved on the same database during a 30-minute, steady state run. All required transaction statistics were met. See the Auditor's attestation letter for details.

5.11. Measurement Interval Duration

A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included.

The measurement interval was 30 minutes.

5.12. Regulation of Transaction Mix

The method of regulation of the transaction mix (e.g. card decks or weighted random distribution) must be described. If weighted distribution is used and the RTE adjusts the weights associated with each transaction type, the maximum adjustments to the weight from the initial value must be disclosed.

The RTE was given a weighed random distribution which could not be adjusted during the run.

5.13. Transaction Statistics

The percentage of the total mix for each transaction type must be disclosed.

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed.

The average number of order-lines entered per New-Order transaction must be disclosed.

The percentage of remote order-lines entered per New-Order transaction must be disclosed.

The percentage of remote Payment transactions must be disclosed.

The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed.

The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

Table 5.4 shows this information.

Table 5.4: Transaction Statistics

Transaction Type	Statistics	Value
New Order	Rolledback transactions	1.01%
	Home warehouse	98.99%
	Remote warehouse	1.01%
	Average Items per Order	9.99
Payment	Home warehouse	84.96%
	Remote warehouse	15.04%
	Non-primary key access	60.03%
Order Status	Non-primary key access	60.10%
Delivery	Skipped transactions (Interactive)	0
	Skipped transaction counts (Deferred)	0
	Skipped District counts (Deferred)	0
Transaction Mix	New Order	44.75%
	Payment	43.17%
	Delivery	4.04%
	Stock-Level	4.03%
	Order-Status	4.01%

5.14. Checkpoint Statistics

The number of checkpoints in the measurement interval, the time in seconds from the start of the measurement interval to the first checkpoint, and the Checkpoint Interval must be disclosed.

There is one checkpoint in the measurement interval. The checkpoint starts 671 seconds into the measurement interval. The checkpoint interval is 30 minutes (from the start of one to the start of the next) and a checkpoint lasts approximately 6.5 minutes. In conformance with Clause 5.2.2 there is no checkpoint within a span of 7.5 minutes before or after the beginning or end of the measurement interval.

6. Clause 6: SUT, Driver & Communications Definition

6.1. Remote Terminal Emulator (RTE) Description

The RTE input parameters, code fragments, functions, etc. used to generate each transaction input field must be disclosed.

The RTE used is proprietary to Unisys. Appendix D contains the profile used as input to this RTE.

6.2. Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system.

There were no emulated components in the benchmark configuration other than the emulated web browsers on the users' PCs.

6.3. Functional Diagrams

A complete functional diagram of both benchmark and the configuration of the proposed (target) system must be disclosed. A detailed list of all hardware and software functionality being performed on the Driver System and its interface to the SUT must be disclosed.

Section 0.7 describes and shows functional diagrams of the benchmarked and priced systems.

6.4. Network Configuration

The network configuration of both the tested and proposed (target) services which are being represented and a thorough explanation of exactly which parts are being replaced with the Driver System must be disclosed.

Figures 0.1 and 0.2 in Section 0.7 also diagram the network configurations of the benchmark and configured systems and represent the RTEs connected via LAN replacing the user PCs that are directly connected via LAN.

6.5. Network Bandwidth

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed.

Ethernet local area networks (LAN) with a bandwidth of 10 megabits per second are used in the tested/priced configurations between RTE/emulated web browsers and the client machines. A single Ethernet LAN with a bandwidth of 100 megabits per second is used between the client machines and the database server (SUT).

Each of the clients contains one 100 megabit per second LAN adapter and two 10 megabit per second LAN adapters. The 100 megabit per second LAN adapter is connected to a single LAN segment and to the database server in both priced and tested configurations.

In the priced configuration, the clients are each connected via two 10Mbit LAN segments to workstations (PCs running web browsers).

In the tested configuration, each client also contains two 10 megabit per second LAN adapters. Each LAN adapter connects to a LAN adapter on an RTE (driver) machine.

6.6. Operator Intervention

If the configuration requires operator intervention (see Clause 6.6.6), the mechanism and the frequency of this intervention must be disclosed.

No operator intervention was required to sustain eight hours of operation at the reported throughput.

7. Clause 7: Pricing

7.1. Pricing

A detailed list of hardware and software used in the priced system must be reported. Each separately orderable item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, vendor part number of the package and a description uniquely identifying each of the components of the package must be disclosed. Pricing source(s) and effective date(s) must also be reported.

The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

System pricing should include subtotals for the following components: Server Hardware, Server Software, Client Hardware, Client Software, and Network Components used for terminal connection (see Clause 7.2.2.3). Clause 6.1 describes the Server and Client components.

System pricing must include line item indication where non-sponsoring companies' brands are used. System pricing must also include line item indication of third party pricing.

A detailed list of hardware and software components along with their part numbers and prices are given in the Executive Summary near the beginning of this document.

7.1.1. System Pricing

Each priced configuration consists of an integrated system package, additional options, and components. Prices for all products are US list prices. A three year warranty is standard with this class of Unisys server products.

7.1.2. Maintenance Pricing

The five year support pricing for Unisys Corporation Open Business Server products is based on a 36-month warranty on hardware and 24 months of monthly support. Microsoft and BEA support pricing is based on 60 months of monthly support costs.

Unisys Corporation Standard Performance-Gold Support: four hour maximum response, onsite support for hardware provides service from 8:00 A.M. to 5:00 P.M., Monday through Friday. Service requests made as late as 5:00 P.M. will receive a response the same day.

Western Micro, Netlux and ALR provide return-to-factory replacement within seven days. Appropriate spares are included in the priced configuration.

7.1.3. Discounts

No discounts were applied to the priced configuration.

7.2. Availability

The committed delivery date for general availability (availability date) of products used in the price calculation must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

The hardware, software and support/maintenance products priced in this benchmark are detailed on page vi.

All components will be available on January 30th, 1998.

7.3. Measured tpmC, Price/Performance, and Availability Date

A statement of the measured tpmC as well as the respective calculations for the 5-year pricing, price/performance (price/tpmC), and the availability date must be included.

Unisys Corporation Aquanta HS/6 Server, with Microsoft Windows NT Server 4.0 Enterprise Edition and SQL Server 6.5 Enterprise Edition, achieved 13,089.30 tpmC at $37.96 per tpmC. All components will be available by January 30th, 1998.

7.4. Country-Specific Pricing

Additional Clause 7 related items may be included in the Full Disclosure Report for each country specific priced configuration. Country specific pricing is subject to Clause 7.1.7.

None.

7.5. Usage Pricing

For any usage pricing, the sponsor must disclose:

- *Usage level at which the component was priced.*
- *A statement of the company policy allowing such pricing.*

The component pricing based on usage is shown below:

- One (1) Microsoft Windows NT Server 4.0 Enterprise Edition license
- One (1) Microsoft SQL 6.50 Server Enterprise Edition license
- Five (5) Microsoft Windows NT 4.0 Licenses
- One (1) Microsoft SQL Server Programmers Toolkit
- One (1) Microsoft Visual C++ Subscription
- Five (5) BEA Tuxedo 6.3 CFS for NT licenses

Microsoft SQL Server & Internet Information Server and BEA Tuxedo were priced for an unlimited number of users.

8. Clause 8 : Full Disclosure Availability

8.1. Availability

The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to charges for similar documents by that test sponsor.

Copies of this Full Disclosure Report may be obtained by contacting:

TPC Benchmark Administrator
Systems Analysis, Modeling & Measurement Group
Unisys Corporation, M/S 262
25725 Jeronimo Road
Mission Viejo, CA 92691
USA

9. *Clause 9 : Audit*

9.1. Auditor's Report

The auditor's name, address, phone number and a copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report.

This implementation of the TPC Benchmark C on the Unisys Aquanta HS/6 Server was audited by Richard Gimarc, a TPC certified auditor of:

Performance Metrics Inc.,
2229 Benita Drive, Suite 101,
Rancho Cordova, CA 95670.

(916)635-2822 Fax: (916) 858-0109
e-mail: Richard@PerfMetrics.com

The attestation letter is shown on the next page.

PERFORMANCE METRICS INC.
TPC Certified Auditors

Jerrold Buggert
Director of Modeling and Measurement
Unisys Corporation
25725 Jeronimo
Mission Viejo, CA 92691

October 28, 1997

I have verified the TPC Benchmark™ C for the following configuration:

Platform: Unisys Aquanta HS/6 Server
Database Manager: Microsoft SQL Server 6.5 Enterprise Edition (6.50.258)
Operating System: Microsoft Windows NT Server 4.0 Enterprise Edition
Transaction Manager: BEA Tuxedo CFS 6.3 for NT

CPUs	Memory	Disks	New-Order Response Time @ 90%	tpmC
Server: Unisys Aquanta HS/6 Server				
6 Pentium Pro @ 200 MHz	Main: 4 GB L2 Cache: 1 MB	88 @ 4.05 GB 62 @ 8.48 GB	0.78 sec.	13,089.30
6 Clients: Unisys Aquanta DM/6 II				
1 Pentium Pro @ 200 MHz	Main: 256 MB L2 Cache: 256 KB	1 @ 1.51 GB	n.a.	n.a.

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark.

The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 1,110 warehouses. Only 1,065 warehouses were used during measurement.
- The ACID properties were met, including phantom protection.
- The durability data loss and log loss tests were performed on a 10-warehouse database.
- Input data was generated according to the specified percentages.

2229 Benita Drive, Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 Fax: (916) 858-0109 e-mail: Richard@PerfMetrics.com

PERFORMANCE METRICS INC.
TPC Certified Auditors

- Eight hours of mirrored log space was configured on the measured system.
- The following disks contained backup and other data and were not active during measurement: six 4.05 GB and fourteen 8.48 GB disks. These twenty disks were not included in the priced configuration.
- The 180-day space calculation was verified. Nine 8.48 GB disks were added to the priced configuration to satisfy this requirement.
- Measurement cycle times include a 0.1 second menu and a 0.1 second response time delay for an emulated Web browser.
- The steady state portion of the test was 30 minutes.
- One checkpoint was taken during the steady state portion of the test.
- Checkpoints were verified to be clear of the guard zones.
- There were 10,650 user contexts present on the system.
- Each emulated user started with a different random number seed.
- The NURand constants used for database load and run time were verified.
- System pricing was checked for major components and maintenance.

Additional Audit Notes: (none)

Regards,

Richard L. Gimarc

Richard L. Gimarc
Auditor

2229 Benita Drive, Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 Fax: (916) 858-0109 e-mail: Richard@PerfMetrics.com

Appendix A - Client/Server Source

CLIENT MAKEFILE

```
# Microsoft Developer Studio Generated NMAKE File, Format Version 4.20
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

!IF "$(CFG)" == ""
CFG=tpcc - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to tpcc - Win32 Debug.
!ENDIF

!IF "$(CFG)" != "tpcc - Win32 Release" && "$(CFG)" != "tpcc - Win32 Debug"
!MESSAGE Invalid configuration "$(CFG)" specified.
!MESSAGE You can specify a configuration when running NMAKE on this
makefile
!MESSAGE by defining the macro CFG on the command line.  For example:
!MESSAGE
!MESSAGE NMAKE /f "tpcc.mak" CFG="tpcc - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "tpcc - Win32 Release" (based on "Win32 (x86) Dynamic-Link
Library")
!MESSAGE "tpcc - Win32 Debug" (based on "Win32 (x86) Dynamic-Link
Library")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
################################################################################
######
# Begin Project
# PROP Target_Last_Scanned "tpcc - Win32 Release"
CPP=cl.exe
RSC=rc.exe
MTL=mktyplib.exe

!IF  "$(CFG)" == "tpcc - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
OUTDIR=.\Release
INTDIR=.\Release

ALL : "$(OUTDIR)\tpcc.dll"

CLEAN :
       -@erase "$(INTDIR)\term.obj"
       -@erase "$(INTDIR)\timesupp.obj"
       -@erase "$(INTDIR)\TPCC.OBJ"
       -@erase "$(INTDIR)\tpcchandler.obj"
       -@erase "$(OUTDIR)\tpcc.dll"
       -@erase "$(OUTDIR)\tpcc.exp"
       -@erase "$(OUTDIR)\tpcc.lib"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D
"_WINDOWS" /YX /c
# ADD CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /YX
/c
CPP_PROJ=/nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS"\
 /Fp"$(INTDIR)/tpcc.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\Release/
CPP_SBRS=.\.
# ADD BASE MTL /nologo /D "NDEBUG" /win32
# ADD MTL /nologo /D "NDEBUG" /win32
MTL_PROJ=/nologo /D "NDEBUG" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/tpcc.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib libtux.lib libbuft.lib libtux2.lib libfml.lib libfml32.lib
libgp.lib tmon.lib /nologo /subsystem:windows /dll /machine:I386
# SUBTRACT LINK32 /verbose /nodefaultlib
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib libtux.lib libbuft.lib libtux2.lib libfml.lib libfml32.lib\
 libgp.lib tmon.lib /nologo /subsystem:windows /dll /incremental:no\
 /pdb:"$(OUTDIR)/tpcc.pdb" /machine:I386 /def:".\tpcc.def"\
 /out:"$(OUTDIR)/tpcc.dll" /implib:"$(OUTDIR)/tpcc.lib"
DEF_FILE= \
       ".\tpcc.def"
LINK32_OBJS= \
```

```
        "$(INTDIR)\term.obj" \
        "$(INTDIR)\timesupp.obj" \
        "$(INTDIR)\TPCC.OBJ" \
        "$(INTDIR)\tpcchandler.obj"

"$(OUTDIR)\tpcc.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : "$(OUTDIR)\tpcc.dll"

CLEAN :
        -@erase "$(INTDIR)\term.obj"
        -@erase "$(INTDIR)\timesupp.obj"
        -@erase "$(INTDIR)\TPCC.OBJ"
        -@erase "$(INTDIR)\tpcchandler.obj"
        -@erase "$(INTDIR)\vc40.idb"
        -@erase "$(INTDIR)\vc40.pdb"
        -@erase "$(OUTDIR)\tpcc.dll"
        -@erase "$(OUTDIR)\tpcc.exp"
        -@erase "$(OUTDIR)\tpcc.ilk"
        -@erase "$(OUTDIR)\tpcc.lib"
        -@erase "$(OUTDIR)\tpcc.pdb"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_WINDOWS" /YX /c
# ADD CPP /nologo /MT /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_WINDOWS" /YX /c
CPP_PROJ=/nologo /MT /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_WINDOWS"\
 /Fp"$(INTDIR)/tpcc.pch" /YX /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
CPP_OBJS=.\Debug/
CPP_SBRS=.\.
# ADD BASE MTL /nologo /D "_DEBUG" /win32
# ADD MTL /nologo /D "_DEBUG" /win32
MTL_PROJ=/nologo /D "_DEBUG" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/tpcc.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib libtux.lib libbuft.lib libtux2.lib libfml.lib libfml32.lib
libgp.lib tmon.lib /nologo /subsystem:windows /dll /debug /machine:I386
# SUBTRACT LINK32 /verbose /nodefaultlib
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib libtux.lib libbuft.lib libtux2.lib libfml.lib libfml32.lib\
 libgp.lib tmon.lib /nologo /subsystem:windows /dll /incremental:yes\
 /pdb:"$(OUTDIR)/tpcc.pdb" /debug /machine:I386 /def:".\tpcc.def"\
 /out:"$(OUTDIR)/tpcc.dll" /implib:"$(OUTDIR)/tpcc.lib"
DEF_FILE= \
        ".\tpcc.def"
LINK32_OBJS= \
        "$(INTDIR)\term.obj" \
        "$(INTDIR)\timesupp.obj" \
        "$(INTDIR)\TPCC.OBJ" \
        "$(INTDIR)\tpcchandler.obj"

"$(OUTDIR)\tpcc.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

################################################################################
######
# Begin Target

# Name "tpcc - Win32 Release"
# Name "tpcc - Win32 Debug"

!IF  "$(CFG)" == "tpcc - Win32 Release"

!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
```

```
!ENDIF

################################################################################
######
# Begin Source File

SOURCE=.\term.c
DEP_CPP_TERM_=\
        ".\term.h"\
        ".\timesupp.h"\
        ".\diagio.h"\


"$(INTDIR)\term.obj" : $(SOURCE) $(DEP_CPP_TERM_) "$(INTDIR)"


# End Source File
################################################################################
######
# Begin Source File

SOURCE=.\timesupp.c
DEP_CPP_TIMES=\
        ".\timesupp.h"\


"$(INTDIR)\timesupp.obj" : $(SOURCE) $(DEP_CPP_TIMES) "$(INTDIR)"


# End Source File
################################################################################
######
# Begin Source File

SOURCE=.\TPCC.C
DEP_CPP_TPCC_=\
        ".\term.h"\
        ".\tpcc.h"\
        ".\tpcchandler.h"\
        ".\diagio.h"\
        {$(INCLUDE)}"\SQLDB.H"\
        {$(INCLUDE)}"\SQLFRONT.H"\
        {$(INCLUDE)}"\tmon.h"\


"$(INTDIR)\TPCC.OBJ" : $(SOURCE) $(DEP_CPP_TPCC_) "$(INTDIR)"


# End Source File
################################################################################
######
# Begin Source File

SOURCE=.\tpcchandler.c
DEP_CPP_TPCCH=\
        ".\term.h"\
        ".\tpcc.h"\
        ".\tpcchandler.h"\
        ".\diagio.h"\
        {$(INCLUDE)}"\SQLDB.H"\
        {$(INCLUDE)}"\SQLFRONT.H"\
        {$(INCLUDE)}"\tmon.h"\


"$(INTDIR)\tpcchandler.obj" : $(SOURCE) $(DEP_CPP_TPCCH) "$(INTDIR)"


# End Source File
################################################################################
######
# Begin Source File

SOURCE=.\tpcc.def

!IF  "$(CFG)" == "tpcc - Win32 Release"

!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"

!ENDIF

# End Source File
# End Target
# End Project
################################################################################
######
```

tpcc.def

```
EXPORTS
   GetExtensionVersion
   HttpExtensionProc
```

tpcc.h

```
// tpcc.h

#include <time.h>
#define DBNTWIN32
#include <sqlfront.h>
#include <sqldb.h>

// TPCCHandler return codes
#define TPCCSEND 1
#define TPCCSENDEND 2
#define TPCCENDNOW 3

// TPCC Service return codes
#define SVC_BADITEMID 1
#define SVC_NOERROR 0
#define SVCERR_DEADLOCK -1
#define SVCERR_NOCUSTOMER -2
#define SVCERR_NOORDERS -3
#define SVCERR_DBLIB -4

// Min/Max transaction data definitions
#define MIN_DId 1
#define MAX_DId 10
#define MIN_OL 5
```

```
#define MAX_OL 15
#define MIN_QUANTITY 1
#define MAX_QUANTITY 10
#define MIN_ITEM_ID 1
#define MAX_ITEM_ID 100000
#define MIN_CUST_ID 1
#define MAX_CUST_ID 3000
#define MIN_CARRIER 1
#define MAX_CARRIER 10
#define MIN_THRESHOLD 10
#define MAX_THRESHOLD 20

// pTPCC->iStatusId codes
#define INVALID_IID              1
#define STATUS_OK                0
#define ERR_CMD_UNKNOWN          -10
#define ERRTXT_CMD_UNKNOWN       "Unrecognized Command"
#define ERR_ALREADY_LOGGEDIN     -11
#define ERRTXT_ALREADY_LOGGEDIN  "Already Logged In"
#define ERR_TERMID               -12
#define ERRTXT_TERMID            "TermId or SyncId in Error"
#define ERR_FORM_UNKNOWN         -13
#define ERRTXT_FORM_UNKNOWN      "Unrecognized FormId"
#define ERR_WID_INVALID          -14
#define ERR_DID_INVALID          -15
#define ERR_MISSING_KEY          -16
#define ERR_NOT_NUMERIC          -17
#define ERR_THRESHOLD_RANGE      -18
#define ERR_EMBEDDED_EMPTY_OL    -19
#define ERR_QUANTITY_INVALID     -20
#define ERR_OL_INVALID           -21
#define ERR_OL_COUNT             -22
#define ERR_TM_INTERFACE         -23
#define ERR_SERVICE_RSLT         -24
#define ERR_INPUT_TOOLONG        -25
#define ERR_IDANDNAME_EMPTY      -26
#define ERR_IDANDNAME_ENTERED    -27
#define ERR_AMOUNT_BADFORM       -28
#define ERR_AMOUNT_INVALID       -29
#define ERR_CARRIER_INVALID      -30
#define ERR_TERM_ALLOC           -31

#define STATUS_LEN 200
#define NAME_LEN 16
#define ADDR_LEN 20
#define STATE_LEN 2
#define ZIP_LEN 9

#define MAX_MSG_SZ  5000

typedef struct
{
   short ol_supply_w_id;
   long ol_i_id;
   char ol_i_name[25];
   short ol_quantity;
   char ol_brand_generic[2];
   double ol_i_price;
   double ol_amount;
   short ol_stock;
} OL_NEW_ORDER_DATA;

typedef struct
{
   short w_id;
   short d_id;
   long        c_id;
   short o_ol_cnt;
   char c_last[NAME_LEN + 1];
   char c_credit[3];
   double c_discount;
   double w_tax;
   double d_tax;
   long o_id;
   short o_commit_flag;
   DBDATEREC o_entry_d;
   short o_all_local;
   double total_amount;
   char execution_status[STATUS_LEN];
   OL_NEW_ORDER_DATA Ol[MAX_OL];
} NEW_ORDER_DATA;

typedef struct
{
   short w_id;
   short d_id;
   long c_id;
   short c_d_id;
   short c_w_id;
   double h_amount;
   DBDATEREC h_date;
   char w_street_1[ADDR_LEN + 1];
   char w_street_2[ADDR_LEN + 1];
   char w_city[ADDR_LEN + 1];
   char w_state[STATE_LEN + 1];
   char w_zip[ZIP_LEN + 1];
   char d_street_1[ADDR_LEN + 1];
   char d_street_2[ADDR_LEN + 1];
   char d_city[ADDR_LEN + 1];
   char d_state[STATE_LEN + 1];
   char d_zip[ZIP_LEN + 1];
   char c_first[NAME_LEN + 1];
   char c_middle[3];
   char c_last[NAME_LEN + 1];
   char c_street_1[ADDR_LEN + 1];
   char c_street_2[ADDR_LEN + 1];
   char c_city[ADDR_LEN + 1];
   char c_state[STATE_LEN + 1];
   char c_zip[ZIP_LEN + 1];
   char c_phone[16];
   DBDATEREC c_since;
   char c_credit[3];
   double c_credit_lim;
   double c_discount;
   double c_balance;
   char c_data[200+1];
   char execution_status[STATUS_LEN];
} PAYMENT_DATA;

typedef struct
{
   long ol_i_id;
```

```
   short ol_supply_w_id;
   short ol_quantity;
   double ol_amount;
   DBDATEREC ol_delivery_d;
} OL_ORDER_STATUS_DATA;

typedef struct
{
   short w_id;
   short d_id;
   long c_id;
   char c_first[NAME_LEN + 1];
   char c_middle[3];
   char c_last[NAME_LEN + 1];
   double c_balance;
   long o_id;
   DBDATEREC o_entry_d;
   short o_carrier_id;
   OL_ORDER_STATUS_DATA OlOrderStatusData[MAX_OL];
   short o_ol_cnt;
   char execution_status[STATUS_LEN];
} ORDER_STATUS_DATA;

typedef struct
{
   short w_id;
   short o_carrier_id;
   long o_id[10];
   int iComplete;
   SYSTEMTIME QTime;               // time delivery was queued
   SYSTEMTIME EndTime;             // time delivery completed
   char execution_status[STATUS_LEN];
} DELIVERY_DATA;

typedef struct
{
   short w_id;
   short d_id;
   short thresh_hold;
   long low_stock;
   char execution_status[STATUS_LEN];
} STOCK_LEVEL_DATA;

typedef struct
{
   LPVOID ConnID;                  // Active Connection Id
   SHORT sWId;                     // TPCC WareHouse Id
   SHORT sDId;                     // TPCC District Id
   INT  iSyncId;                   // TPCC Sync Id
   INT  iTermId;                   // TPCC Term Id
   UINT uFormId;                   // TPCC Form Id
   INT  iStatusId;                 // TPCC Status Id
   CHAR ErrTxt[500];               // Error text
   CHAR szWork[200];               // Thread work area
   CHAR szHeader[100];             // HTTP work area
   CHAR * RecvMsg;                 // HTML message from ECB
   CHAR SendMsg[MAX_MSG_SZ];       // HTML work area
   TMON_STATE tsTMon;              // TMon Interface
} TPCC_STATE;
```

tpcc.c

```
// tpcc.c
//
// Copyright Unisys, 1997
//
#include <windows.h>
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <winreg.h>
#include <httpext.h>
#include <tmon.h>

#include "diagio.h"
#include "term.h"
#include "tpcchandler.h"

#define EXTN_VERSION MAKELONG(HSE_VERSION_MINOR,HSE_VERSION_MAJOR)
#define TLS_NULL 0xFFFFFFFF
DWORD dwTlsInx;
CHAR * pTitle = "IIS TPCC DLL";
CRITICAL_SECTION csDllMain;

// Diagnostic logging settings
BOOL bEventLog = TRUE;
BOOL bConsole = FALSE;
UINT uDiagLevel = DIAG_INFO;

// TMon Interface Settings
INT iTMSyncWait = DEFAULTSYNCWAIT;
INT iTMMaxMsg = 0;

// Term Interface Settings
INT iMaxTerms = 3000;

static CHAR * szTPCCError =
   "<HTML>"
   "<HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY>"
   "<B>TPCC Extension Error (TPCC Array Not Allocated)</B><BR>"
   "</BODY></HTML>";

static CHAR * szTMInitError =
   "<HTML>"
   "<HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY>"
   "<B>TPCC Extension Error (TMInit Failed)</B><BR>"
   "</BODY></HTML>";

BOOL ThreadAttach(TPCC_STATE * pTPCC,CHAR * pDiag);
VOID ThreadDetach(TPCC_STATE * pTPCC);
VOID SendResponse(EXTENSION_CONTROL_BLOCK * pECB,CHAR * pMsg,CHAR *
pWork);
BOOL ReadRegistry(VOID);

//============================================================================
//
// Function name: DllMain
//
//============================================================================
```

```
BOOL APIENTRY DllMain(HANDLE hInst, ULONG ul_reason_for_call,
                      LPVOID lpReserved)
{
   TPCC_STATE * pTPCC = NULL;
   CHAR szDiag[MAX_DIAG_SZ];
   UINT iTMMaxSz =0;
   switch(ul_reason_for_call)
   {
      case DLL_PROCESS_ATTACH:
         // Process initialization

         InitializeCriticalSection(&csDllMain);
         ReadRegistry();
         DiagIoInit(pTitle,bConsole,bEventLog,uDiagLevel);
         sprintf(szDiag,
            "EventLog = %d, Console = %d, DiagLevel = %d\n"
            "SyncWait = %d, MaxTerms = %d\n",
            bEventLog,bConsole,uDiagLevel,iTMSyncWait,iMaxTerms);
         DiagIoWrite(szDiag,DIAG_FORCE);
         dwTlsInx = TlsAlloc();
         if (dwTlsInx == TLS_NULL)
         {
            sprintf(szDiag,"PAttach(%ld): Tls Alloc Failed (%ld)\n",
               GetCurrentThreadId(),GetLastError);
            DiagIoWrite(szDiag,DIAG_ERROR);
            return(FALSE);
         };
         if (TermInit(iMaxTerms))
            return(FALSE);
         iTMMaxSz = max(iTMMaxSz,sizeof(NEW_ORDER_DATA));
         iTMMaxSz = max(iTMMaxSz,sizeof(PAYMENT_DATA));
         iTMMaxSz = max(iTMMaxSz,sizeof(ORDER_STATUS_DATA));
         iTMMaxSz = max(iTMMaxSz,sizeof(DELIVERY_DATA));
         iTMMaxSz = max(iTMMaxSz,sizeof(STOCK_LEVEL_DATA));
         iTMMaxSz += 10;
         TMonInit(iTMMaxSz,iTMSyncWait);
         break;
      case DLL_THREAD_ATTACH:
         // Move ThreadAttach call to HttpExt since the DllMain call
         // for Thread Attach did not reliably come before the first
         // call to HttpExtProc.
         break;
               case DLL_THREAD_DETACH:
         ThreadDetach(pTPCC);
                     break;
               case DLL_PROCESS_DETACH:
         ThreadDetach(pTPCC);
         DeleteCriticalSection(&csDllMain);
         TMonTerm();
         TermTerm();
         TlsFree(dwTlsInx);
         dwTlsInx = TLS_NULL;
         DiagIoTerm();
                     break;
       };
       return TRUE;
};   // DllMain

//=========================================================================
//
// Function name: ThreadAttach
//
// Result:
//    FALSE Thread state structure initialized
//    TRUE  Thread state structure initialization failure
//
//=========================================================================
BOOL ThreadAttach(TPCC_STATE * pTPCC,CHAR * pDiag)
{
   BOOL bRslt;
   UINT uLabelNoOp;
   EnterCriticalSection(&csDllMain);
   __try
   {
      pTPCC = (TPCC_STATE *) calloc(1,sizeof(TPCC_STATE));
      if (pTPCC == NULL)
      {
         sprintf(pDiag,"ThrAtt(%ld): pTPCC Alloc Failed (%ld)\n",
            GetCurrentThreadId(),GetLastError);
         DiagIoWrite(pDiag,DIAG_ERROR);
         bRslt = TRUE;
         goto TAttachXit;
      };
      TlsSetValue(dwTlsInx,pTPCC);
      pTPCC->tsTMon.pTMData = NULL;
      pTPCC->tsTMon.pszErrTxt = pTPCC->ErrTxt;
      if (TMInit(&pTPCC->tsTMon))
      {
         sprintf(pDiag,"ThrAtt(%ld): TMInit %s\n",
            GetCurrentThreadId(),pTPCC->ErrTxt);
         DiagIoWrite(pDiag,DIAG_ERROR);
         bRslt = TRUE;
         goto TAttachXit;
      };
      bRslt = FALSE;
TAttachXit:
      uLabelNoOp = 0;
   }
   __finally
   {
      LeaveCriticalSection(&csDllMain);
   };

   return(bRslt);

};   // ThreadAttach

//=========================================================================
//
// Function name: ThreadDetach
//
//=========================================================================
VOID ThreadDetach(TPCC_STATE * pTPCC)
{
   EnterCriticalSection(&csDllMain);
   __try
   {
      pTPCC = TlsGetValue(dwTlsInx);
      if (pTPCC != NULL)
      {
         TMDone(&pTPCC->tsTMon);
         free(pTPCC);
```

```
         pTPCC = NULL;
         TlsSetValue(dwTlsInx,pTPCC);
      };
   }
   __finally
   {
      LeaveCriticalSection(&csDllMain);
   };
};   // ThreadDetach

//=========================================================================
//
// Function name: GetExtensionVersion
//
//=========================================================================
BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVersion)
{
   pVersion->dwExtensionVersion = EXTN_VERSION;
   strncpy(pVersion->lpszExtensionDesc,pTitle,HSE_MAX_EXT_DLL_NAME_LEN);
   return TRUE;
};   // GetExtensionVersion

//=========================================================================
//
// Function name: HttpExtensionProc
//
//
// Returns:
//    HSE_STATUS_SUCCESS               send msg, drop connection
//    HSE_STATUS_SUCCESS_AND_KEEP_CONN       send msg, keep connection
//
//=========================================================================
DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK * pECB)
{
   TPCC_STATE * pTPCC;
   DWORD dwRslt = HSE_STATUS_SUCCESS;
   UINT uRslt;

   pTPCC = TlsGetValue(dwTlsInx);
   if (pTPCC == NULL)
   {
      CHAR szWork[200];
      ThreadAttach(pTPCC,szWork);
      pTPCC = TlsGetValue(dwTlsInx);
      if (pTPCC == NULL)
      {
         SendResponse(pECB,szTPCCError,szWork);
         goto HttpXit;
      };
   };
   if (pTPCC->tsTMon.pTMData == NULL)
      SendResponse(pECB,szTMInitError,pTPCC->szHeader);
   TPCCClear(pTPCC);
   pTPCC->ConnID = pECB->ConnID;
   pTPCC->RecvMsg = pECB->lpszQueryString;
   uRslt = TPCCHandler(pTPCC);
   switch (uRslt)
   {
      case TPCCSEND:
         SendResponse(pECB,pTPCC->SendMsg,pTPCC->szHeader);
         dwRslt = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
         break;
      case TPCCSENDEND:
         SendResponse(pECB,pTPCC->SendMsg,pTPCC->szHeader);
         break;
      case TPCCENDNOW:
      default:
         break;
   };   // switch (TPCCHandler result)

HttpXit:

       return(dwRslt);

};   // HttpExtensionProc

//=========================================================================
//
// Function name: SendResponse
//
//=========================================================================
VOID SendResponse(EXTENSION_CONTROL_BLOCK * pECB,CHAR * pMsg,CHAR * pWork)
{
   DWORD dwHdrBytes;
   DWORD dwMsgBytes;
   dwMsgBytes = strlen(pMsg);
   sprintf(pWork,
      "Connection: keep-alive\r\nContent-type: text/html\r\n"
      "Content-length: %d\r\n\r\n",
      dwMsgBytes);
   dwHdrBytes = strlen(pWork);
   (*pECB->ServerSupportFunction)
      (pECB->ConnID,
      HSE_REQ_SEND_RESPONSE_HEADER,
      NULL,
      &dwHdrBytes,
      (LPDWORD)pWork);
   (*pECB->WriteClient)
      (pECB->ConnID,
      pMsg,
      &dwMsgBytes,
      0);
};   // SendResponse

//=========================================================================
//
// Function name: ReadRegistry
//
//    Sets global operational parameters from registry if they exist.
//    Otherwise, compiled in defaults apply.
//
// Result:
//    FALSE    Registry entry found
//    TRUE     Registry entry does not exist
//
//=========================================================================
BOOL ReadRegistry(VOID)
{
   HKEY hkTPCC;
   DWORD       dwMax;
```

```
   DWORD       dwRT;
   INT i;
   CHAR szValue[100];
   if (RegOpenKeyEx(HKEY_LOCAL_MACHINE,"SOFTWARE\\Unisys\\TPCC",0,
      KEY_READ, &hkTPCC) != ERROR_SUCCESS )
      return(TRUE);
   dwMax = sizeof(szValue);
   if (RegQueryValueEx(hkTPCC,"EVENTLOG",0,&dwRT,szValue,&dwMax)
      == ERROR_SUCCESS)
   {
      if (abs(atoi(szValue) == 0))
         bEventLog = FALSE;
      else
         bEventLog = TRUE;
   };
   dwMax = sizeof(szValue);
   if (RegQueryValueEx(hkTPCC,"CONSOLE",0,&dwRT,szValue,&dwMax)
      == ERROR_SUCCESS )
   {
      if (abs(atoi(szValue) == 0))
         bConsole = FALSE;
      else
         bConsole = TRUE;
   };
   dwMax = sizeof(szValue);
   if (RegQueryValueEx(hkTPCC,"DIAGLEVEL",0,&dwRT,szValue,&dwMax)
      == ERROR_SUCCESS )
   {
      i = atoi(szValue);
      if (i < DIAG_FORCE)
         i = DIAG_FORCE;
      else
      if (i > DIAG_INFO)
         i = DIAG_INFO;
      uDiagLevel = i;
   };
   dwMax = sizeof(szValue);
   if (RegQueryValueEx(hkTPCC,"SYNCWAIT",0,&dwRT,szValue,&dwMax)
      == ERROR_SUCCESS )
   {
      iTMSyncWait = abs(atoi(szValue));
   };
   dwMax = sizeof(szValue);
   if (RegQueryValueEx(hkTPCC,"MAXTERMS",0,&dwRT,szValue,&dwMax)
      == ERROR_SUCCESS )
   {
      iMaxTerms = abs(atoi(szValue));
   };
   RegCloseKey(hkTPCC);
   return(FALSE);
};   // ReadRegistry
```

tpcchandler.h

```
// tpcchandler.h

#include "tpcc.h"

BOOL TPCCClear(TPCC_STATE * pTPCC);
UINT TPCCHandler(TPCC_STATE * pTPCC);
```

tpcchandler.c

```
// tpcchandler.c
//
// Copyright Unisys, 1997
//
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <tmon.h>

#include "diagio.h"
#include "tpcchandler.h"
#include "term.h"

// pTPCC->iFormId - TPCC forms enumeration.
#define FORM_NULL        0
#define FORM_LOGON       1
#define FORM_MENU        2
#define FORM_NEWORDER    3
#define FORM_PAYMENT     4
#define FORM_DELIVERY    5
#define FORM_ORDERSTATUS 6
#define FORM_STOCKLEVEL  7
#define FORM_EXIT        8
#define FORM_MAX         9

// CMD= HTML Command Enumeration and Name
#define CMD_NULL              0
#define CMD_PROCESS           1
#define CMD_NEWORDER_FORM     2
#define CMD_PAYMENT_FORM      3
#define CMD_DELIVERY_FORM     4
#define CMD_ORDERSTATUS_FORM  5
#define CMD_STOCKLEVEL_FORM   6
#define CMD_EXIT              7
#define CMD_SUBMIT            8
#define CMD_MENU_FORM         9
#define CMD_MAX               10

static CHAR * szCmds[] =
{
   "Unknown",
   "Process",
   "..NewOrder..",
   "..Payment..",
   "..Delivery..",
   "..Order-Status..",
   "..Stock-Level..",
   "..Exit..",
   "Submit",
   "Menu"
};

static CHAR * szFormLogin =
   "<HTML>"
   "<HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY>"
   "Please Identify your Warehouse and District for this session.<BR>"
   "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">"
```

```
   "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">"
   "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"1\">"
   "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"-2\">"
   "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"0\">"
   "Warehouse ID <INPUT NAME=\"w_id\" SIZE=4><BR>"
   "District ID <INPUT NAME=\"d_id\" SIZE=2><BR>"
   "<HR>"
   "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Submit\">"
   "</FORM>";

static CHAR * szMenuList =
   "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">"
   "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Payment..\">"
   "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Delivery..\">"
   "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Order-Status..\">"
   "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Stock-Level..\">"
   "<INPUT TYPE=\"submit\" NAME=\"CMD\"VALUE=\"..Exit..\">";

static CHAR * HTMLTrailer =
   "</BODY></HTML>";

static CHAR * TERMIDTOKEN = "TERMID=";
static CHAR * SYNCIDTOKEN = "SYNCID=";
static CHAR * FORMIDTOKEN = "FORMID=";
static CHAR * STATUSIDTOKEN = "STATUSID=";
static CHAR * CMDTOKEN = "CMD=";
static CHAR * NEWORDER_SERVICE = "NEWORDER";
static CHAR * PAYMENT_SERVICE = "PAYMENT";
static CHAR * ORDERSTATUS_SERVICE = "ORDERSTS";
static CHAR * DELIVERY_SERVICE = "DELIVERY";
static CHAR * STOCKLEVEL_SERVICE = "STOCKLVL";
static CHAR * ZIPPIC = "XXXXX-XXXX";

BOOL ProcessLogin(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
BOOL ProcessForm(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
BOOL ProcessNewOrder(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
BOOL ProcessPayment(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
BOOL ProcessDelivery(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
BOOL ProcessOrderStatus(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
BOOL ProcessStockLevel(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatLogin(CHAR * pMsg,CHAR * pAddText);
BOOL GetHidden(CHAR * pMsg,UINT * uFormId,INT * iSyncId,INT * iTermId);
BOOL GetCmd(CHAR * pMsg,CHAR * pWork,UINT uLen);
BOOL GetLongKey(LONG * lRslt,CHAR * pHTML,CHAR * pKey,TPCC_STATE * pTPCC);
BOOL GetIntKey(INT * iRslt,CHAR * pHTML,CHAR * pKey,TPCC_STATE * pTPCC);
BOOL GetShortKey(SHORT * sRslt,CHAR * pHTML,CHAR * pKey,TPCC_STATE *
pTPCC);
BOOL GetStringKey(CHAR * szRslt,CHAR * pHTML,CHAR * pKey,
                  TPCC_STATE * pTPCC,UINT uMax);
BOOL GetAmountKey(DOUBLE * dRslt,CHAR * pHTML,CHAR * pKey,
                  TPCC_STATE * pTPCC);
BOOL GetKeyValue(CHAR * pHTML,CHAR * pKey,CHAR * pValue,UINT uMax);
VOID FormatLogin(CHAR * pOut,CHAR * pAddText);
VOID FormatMenu(CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatNewOrder(CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatPayment(CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatDelivery(CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatOrderStatus(CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatStockLevel(CHAR * pOut,TPCC_STATE * pTPCC);
VOID FormatFormHdr(CHAR * pOut,CHAR * pTitle,TPCC_STATE * pTPCC);
VOID FormatRespHdr(CHAR * pOut,CHAR * pTitle,TPCC_STATE * pTPCC);
VOID FormatHTMLString(CHAR * pOut,CHAR * pIn,UINT uLen);
VOID FormatString(CHAR * pOut,CHAR * pPic,CHAR * pIn);
VOID UtilStrCpy(CHAR * pDest,CHAR * pSrc,INT n);
BOOL CheckNumeric(CHAR * pNum);

//============================================================================
//
// Function name: TPCCClear
//
//============================================================================
BOOL TPCCClear(TPCC_STATE * pTPCC)
{
   pTPCC->ConnID = 0;
   pTPCC->sWId = 0;
   pTPCC->sDId = 0;
   pTPCC->iSyncId = 0;
   pTPCC->iTermId = -2;
   pTPCC->uFormId = FORM_NULL;
   pTPCC->iStatusId = 0;
   pTPCC->tsTMon.lTMDataLen = 0;
   strcpy(pTPCC->ErrTxt,"");
   return(FALSE);
};   // TPCCClear

//============================================================================
//
// Function name: TPCCHandler
//
//============================================================================
UINT TPCCHandler(TPCC_STATE * pTPCC)
{
   INT iSyncId;
   INT iTermId;
   UINT uCmdId;
   UINT uRslt = TPCCSENDEND;  // default error handling
   TERM_STATE * pTerm;

   pTPCC->iStatusId = STATUS_OK;
   if (GetHidden(pTPCC->RecvMsg,&pTPCC->uFormId,&iSyncId,&iTermId))
   {
      uRslt = TPCCSEND;
      FormatLogin(pTPCC->SendMsg,pTPCC->ErrTxt);
      goto HdlrXit;
   };
   if (iTermId > 0)
   {
      pTerm = TermGet(iTermId);
      if (pTerm == NULL)
      {
         uRslt = TPCCSEND;
         strcpy(pTPCC->ErrTxt,"Invalid Term Id");
         FormatLogin(pTPCC->SendMsg,pTPCC->ErrTxt);
         goto HdlrXit;
      };
      if (pTerm->ConnID != pTPCC->ConnID)
      {
         uRslt = TPCCSEND;
         strcpy(pTPCC->ErrTxt,"TermId vs ConnId Mismatch");
         FormatLogin(pTPCC->SendMsg,pTPCC->ErrTxt);
```

```
      goto HdlrXit;
   };
   pTPCC->sWId = pTerm->sWId;
   pTPCC->sDId = pTerm->sDId;
   pTPCC->iSyncId = pTerm->iSyncId;
   pTPCC->iTermId = pTerm->iTermId;
};
uCmdId = GetCmd(pTPCC->RecvMsg,pTPCC->szWork,sizeof(pTPCC->szWork));
// Except for Submit(log in), sWId must already be set
if (pTPCC->sWId == 0 && uCmdId != CMD_SUBMIT)
{
   strcpy(pTPCC->ErrTxt,"Must log in first!");
   FormatLogin(pTPCC->SendMsg,pTPCC->ErrTxt);
   uRslt = TPCCSEND;
   goto HdlrXit;
};
// Check for multiple log in attempts
if (pTPCC->sWId != 0 && uCmdId == CMD_SUBMIT)
{
   strcpy(pTPCC->ErrTxt,ERRTXT_ALREADY_LOGGEDIN);
   pTPCC->iStatusId = ERR_ALREADY_LOGGEDIN;
   FormatMenu(pTPCC->SendMsg,pTPCC);
   uRslt = TPCCSEND;
   goto HdlrXit;
};
// If not logging in, validate hidden fields
if (uCmdId != CMD_SUBMIT)
{
   if (iTermId != pTPCC->iTermId || iTermId != iSyncId)
   {
      sprintf(pTPCC->ErrTxt,"%s: Received %ld, %ld (%ld)",
         ERRTXT_TERMID,iTermId,iSyncId,pTPCC->iTermId);
      pTPCC->iStatusId = ERR_TERMID;
      FormatMenu(pTPCC->SendMsg,pTPCC);
      goto HdlrXit;
   };
};
// Process the command
switch (uCmdId)
{
   case CMD_SUBMIT:
      ProcessLogin(pTPCC->RecvMsg,pTPCC->SendMsg,pTPCC);
      break;
   case CMD_MENU_FORM:
      FormatMenu(pTPCC->SendMsg,pTPCC);
      break;
   case CMD_PROCESS:
      ProcessForm(pTPCC->RecvMsg,pTPCC->SendMsg,pTPCC);
      break;
   case CMD_NEWORDER_FORM:
      FormatNewOrder(pTPCC->SendMsg,pTPCC);
      break;
   case CMD_PAYMENT_FORM:
      FormatPayment(pTPCC->SendMsg,pTPCC);
      break;
   case CMD_DELIVERY_FORM:
      FormatDelivery(pTPCC->SendMsg,pTPCC);
      break;
   case CMD_ORDERSTATUS_FORM:
      FormatOrderStatus(pTPCC->SendMsg,pTPCC);
      break;
   case CMD_STOCKLEVEL_FORM:
      FormatStockLevel(pTPCC->SendMsg,pTPCC);
      break;
   case CMD_EXIT:
      TermFree(pTPCC->iTermId);
      strcpy(pTPCC->ErrTxt,"Logged Off");
      FormatLogin(pTPCC->SendMsg,pTPCC->ErrTxt);
      goto HdlrXit;
   default:
      strcpy(pTPCC->ErrTxt,ERRTXT_CMD_UNKNOWN);
      pTPCC->iStatusId = ERR_CMD_UNKNOWN;
      if (pTPCC->sWId == 0)
         FormatLogin(pTPCC->SendMsg,pTPCC->ErrTxt);
      else
         FormatMenu(pTPCC->SendMsg,pTPCC);
      break;
};   // switch (uCmdId)

uRslt = TPCCSEND;

HdlrXit:

return(uRslt);

};   // TPCCHandler

//================================================================================
//
// Function name: ProcessLogin
//
//    ProcessLogin extracts WId and DId from the incoming form.  Assumes
//    log in has not previously completed (sWId == 0 already verified).
//
//    Result:
//       FALSE - log in successful, sWId and sDId set in pTPCC,
//               pOut contains menu.
//       TRUE - log in failed, pOut contains log in form with
//              error message.
//
//================================================================================
BOOL ProcessLogin(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   SHORT sWId;
   SHORT sDId;
   TERM_STATE * pTerm;

   if (GetShortKey(&sWId,pIn,"w_id",pTPCC))
   {
     FormatLogin(pOut,pTPCC->ErrTxt);
     return(TRUE);
   };
   if (sWId < 1)
   {
      sprintf(pTPCC->ErrTxt,"Warehouse Id (%d) Invalid",sWId);
      pTPCC->iStatusId = ERR_WID_INVALID;
      FormatLogin(pOut,pTPCC->ErrTxt);
      return(TRUE);
   };
   if (GetShortKey(&sDId,pIn,"d_id",pTPCC))
   {
     FormatLogin(pOut,pTPCC->ErrTxt);
```

```
     return(TRUE);
   };
   if (sDId < MIN_DId || sDId > MAX_DId)
   {
      sprintf(pTPCC->ErrTxt,"DId Out of Range(%ld,%ld) - %ld",
         MIN_DId,MAX_DId,sDId);
      pTPCC->iStatusId = ERR_DID_INVALID;
      FormatLogin(pOut,pTPCC->ErrTxt);
      return(TRUE);
   };
   pTerm = TermAlloc();
   if (pTerm == NULL)
   {
      sprintf(pTPCC->ErrTxt,"Unable to Allocate Terminal Entry");
      pTPCC->iStatusId = ERR_TERM_ALLOC;
      FormatLogin(pOut,pTPCC->ErrTxt);
      return(TRUE);
   };
   pTerm->ConnID = pTPCC->ConnID;
   pTerm->iSyncId = pTerm->iTermId;
   pTerm->sWId = abs(sWId);
   pTerm->sDId = abs(sDId);
   pTPCC->iTermId = pTerm->iTermId;
   pTPCC->iSyncId = pTerm->iSyncId;
   pTPCC->sWId = pTerm->sWId;
   pTPCC->sDId = pTerm->sDId;
   FormatMenu(pOut,pTPCC);
   return(FALSE);
};   // ProcessLogin

//================================================================================
//
// Function name: ProcessForm
//
//    ProcessForm uses pTPCC->uFormId to determine which form input is
//    present and ready for processing.  Actual processing is done by
//    the form specific routine.
//
//    Result:
//       FALSE - form processed, pOut contains response.
//       TRUE - error processing form input, pOut contains reason.
//
//================================================================================
BOOL ProcessForm(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   switch (pTPCC->uFormId )
   {
      case FORM_NEWORDER:
               return(ProcessNewOrder(pIn,pOut,pTPCC));
      case FORM_PAYMENT:
               return(ProcessPayment(pIn,pOut,pTPCC));
      case FORM_DELIVERY:
               return(ProcessDelivery(pIn,pOut,pTPCC));
      case FORM_ORDERSTATUS:
               return(ProcessOrderStatus(pIn,pOut,pTPCC));
      case FORM_STOCKLEVEL:
               return(ProcessStockLevel(pIn,pOut,pTPCC));
      default:
         sprintf(pTPCC->ErrTxt,"%s (%ld)",
            ERRTXT_FORM_UNKNOWN,pTPCC->uFormId);
         pTPCC->iStatusId = ERR_FORM_UNKNOWN;
         FormatMenu(pOut,pTPCC);
         break;
      }
   return(TRUE);
};   // ProcessForm

//================================================================================
//
// Function name: ProcessNewOrder
//
//    ProcessNewOrder extracts the input data fields from pIn, processes
//    the data, and returns a response in pOut.
//
//    Result:
//       FALSE - NewOrder processed successfully.
//       TRUE - NewOrder processing failed.
//
//================================================================================
BOOL ProcessNewOrder(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   NEW_ORDER_DATA * pnod;
   TMON_STATE * pTMon;
   CHAR szKey[20];
   CHAR szCredit[14];
   UINT u;
   BOOL bDone = FALSE;
   BOOL bTMRslt;
   BOOL bTPRslt;
   INT iTPRslt;

   pTMon = &pTPCC->tsTMon;
   pTMon->lTMDataLen = sizeof(NEW_ORDER_DATA);
   memset(pTMon->pTMData,0,pTMon->lTMDataLen);
   pnod = (NEW_ORDER_DATA *) pTMon->pTMData;
   pnod->w_id = pTPCC->sWId;
   if (GetShortKey(&pnod->d_id,pIn,"DID*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   if (pnod->d_id < MIN_DId || pnod->d_id > MAX_DId)
   {
      sprintf(pTPCC->ErrTxt,"DId Out of Range(%ld,%ld) - %ld",
         MIN_DId,MAX_DId,pnod->d_id);
      pTPCC->iStatusId = ERR_DID_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   if (GetLongKey(&pnod->c_id,pIn,"CID*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   pnod->o_ol_cnt = 0;
   for(u=0; u < MAX_OL; u++)
   {
      sprintf(szKey,"IID%2.2d*",u);
      if (GetLongKey(&pnod->Ol[u].ol_i_id,pIn,szKey,pTPCC))
      {
```

```
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
sprintf(szKey,"SP%2.2d*",u);
if (GetShortKey(&pnod->Ol[u].ol_supply_w_id,pIn,szKey,pTPCC))
{
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
sprintf(szKey,"Qty%2.2d*",u);
if (GetShortKey(&pnod->Ol[u].ol_quantity,pIn,szKey,pTPCC))
{
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
if (pnod->Ol[u].ol_i_id != 0)
{
   // Check for prior blank lines
   if (bDone)
   {
      strcat(pTPCC->ErrTxt,"Embedded Empty Order Lines");
      pTPCC->iStatusId = ERR_EMBEDDED_EMPTY_OL;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   if (pnod->Ol[u].ol_supply_w_id < 1)
   {
      sprintf(pTPCC->ErrTxt,
         "Order Line %ld Contains Invalid WId %d",
         u,pnod->Ol[u].ol_supply_w_id);
      pTPCC->iStatusId = ERR_WID_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   if (pnod->Ol[u].ol_quantity < MIN_QUANTITY ||
       pnod->Ol[u].ol_quantity > MAX_QUANTITY)
   {
      sprintf(pTPCC->ErrTxt,
         "Order Line %ld Contains Invalid Qty %d",
         u,pnod->Ol[u].ol_quantity);
      pTPCC->iStatusId = ERR_QUANTITY_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   pnod->o_ol_cnt++;
}   // if (ol_i_id !=0)
else
{
   if (pnod->Ol[u].ol_supply_w_id != 0)
   {
      sprintf(pTPCC->ErrTxt,
         "Order Line %ld WId Supplied with No Item",u);
      pTPCC->iStatusId = ERR_OL_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   if (pnod->Ol[u].ol_quantity != 0)
   {
      sprintf(pTPCC->ErrTxt,
         "Order Line %ld Qty Supplied with No Item",u);
      pTPCC->iStatusId = ERR_OL_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   bDone = TRUE;
};   // empty order line
};   // for (u < MAX_OL)

if (pnod->o_ol_cnt < MIN_OL)
{
   sprintf(pTPCC->ErrTxt,"Too Few Order Lines %d",pnod->o_ol_cnt);
   pTPCC->iStatusId = ERR_OL_COUNT;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
bTMRslt = TMTran(NEWORDER_SERVICE,pTMon,&bTPRslt,&iTPRslt);
pnod = (NEW_ORDER_DATA *) pTMon->pTMData;
if (bTMRslt)
{
   pTPCC->iStatusId = ERR_TM_INTERFACE;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
// Exclude invalid item id case
if (bTPRslt && iTPRslt < SVC_NOERROR)
{
   sprintf(pTPCC->ErrTxt,
      "New Order Service Returned Error(%ld): %s",
      iTPRslt,pnod->execution_status);
   pTPCC->iStatusId = ERR_SERVICE_RSLT;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
if (iTPRslt == SVC_BADITEMID)
   pTPCC->iStatusId = INVALID_IID;

FormatRespHdr(pOut,"TPC-C New Order",pTPCC);
sprintf(pOut + strlen(pOut),
   "<PRE>                                   New Order<BR>"
   "Warehouse: %4.4d   District: %2.2d                          ",
   pnod->w_id,pnod->d_id);
if (!bTPRslt)
   {
   sprintf(pOut + strlen(pOut),
      "Date: %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d <BR>",
      pnod->o_entry_d.day,pnod->o_entry_d.month,
      pnod->o_entry_d.year,pnod->o_entry_d.hour,
      pnod->o_entry_d.minute,pnod->o_entry_d.second);
}
else
{
   sprintf(pOut + strlen(pOut),"Date:<BR>");
};
FormatHTMLString(pTPCC->szWork,pnod->c_last,NAME_LEN);
FormatHTMLString(szCredit,pnod->c_credit,2);
sprintf(pOut + strlen(pOut),
   "Customer:  %4.4d   Name: %s   Credit: %s   ",
   pnod->c_id,pTPCC->szWork,szCredit);
if (!bTPRslt)
{
   sprintf(pOut + strlen(pOut),
      "%%Disc: %5.2f            <BR>",pnod->c_discount * 100);
```

```
      sprintf(pOut + strlen(pOut),
         "Order Number: %8.8d  Number of Lines: %2.2d        W_tax: %5.2f
D_tax: %5.2f  <BR><BR>",
         pnod->o_id,pnod->o_ol_cnt,pnod->w_tax * 100,pnod->d_tax * 100);
      strcat(pOut," Supp_W  Item_Id  Item Name                 Qty  Stock
B/G  Price    Amount<BR>");
      for (u = 0; u < (UINT) pnod->o_ol_cnt; u++)
      {
         FormatHTMLString(pTPCC->szWork,pnod->Ol[u].ol_i_name,24);
         sprintf(pOut + strlen(pOut),
            "  %4.4d   %6.6d   %s  %2.2d    %3.3d    %1.1s   $%6.2f
$%7.2f  <BR>",
            pnod->Ol[u].ol_supply_w_id,pnod->Ol[u].ol_i_id,
            pTPCC->szWork,pnod->Ol[u].ol_quantity,pnod->Ol[u].ol_stock,
            pnod->Ol[u].ol_brand_generic,pnod->Ol[u].ol_i_price,
            pnod->Ol[u].ol_amount );
      }
   }   // if (!bTPRslt)
   else
   {
      strcat(pOut,"%Disc:<BR>");
      sprintf(pOut + strlen(pOut),
         "Order Number: %8.8d  Number of Lines:             W_tax:
D_tax:<BR><BR>",
         pnod->o_id);
      strcat(pOut,
         " Supp_W  Item_Id  Item Name                 Qty  Stock  B/G
Price    Amount<BR>");
      u = 0;
   };
   for(; u < MAX_OL; u++)
      strcat(pOut,"<BR>");
   if (!bTPRslt)
   {
      sprintf(pOut + strlen(pOut),
         "Execution Status: %24.24s                 Total:  $%8.2f  ",
         pnod->execution_status,pnod->total_amount);
   }
   else
   {
      sprintf(pOut + strlen(pOut),
         "Execution Status: %24.24s                 Total:",
         pnod->execution_status);
   };
   sprintf(pOut + strlen(pOut),
      "</PRE><HR><BR>%s</FORM>%s",szMenuList,HTMLTrailer);

   return(FALSE);

};   // ProcessNewOrder

//============================================================================
//
// Function name: ProcessPayment
//
//    ProcessPayment extracts the input data fields from pIn, processes
//    the data, and returns a response in pOut.
//
//    Result:
//       FALSE - Payment processed successfully.
//       TRUE - Payment processing failed.
//
//============================================================================
BOOL ProcessPayment(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   PAYMENT_DATA * ppd;
   TMON_STATE * pTMon;
   BOOL bTMRslt;
   BOOL bTPRslt;
   INT iTPRslt;
   CHAR * pCredit;
   INT iCDLines;
   CHAR szWork2[60];
   CHAR szWork3[60];
   CHAR szWork4[60];
   CHAR szZip1[20];
   CHAR szZip2[20];
   INT i;

   pTMon = &pTPCC->tsTMon;
   pTMon->lTMDataLen = sizeof(PAYMENT_DATA);
   memset(pTMon->pTMData,0,pTMon->lTMDataLen);
   ppd = (PAYMENT_DATA *) pTMon->pTMData;
   ppd->w_id = pTPCC->sWId;
   // Get and validate DId
   if (GetShortKey(&ppd->d_id,pIn,"DID*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   if (ppd->d_id < MIN_DId || ppd->d_id > MAX_DId)
   {
      sprintf(pTPCC->ErrTxt,"DId Out of Range(%ld,%ld) - %ld",
         MIN_DId,MAX_DId,ppd->d_id);
      pTPCC->iStatusId = ERR_DID_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   // Get and validate customer Id and name
   if (GetLongKey(&ppd->c_id,pIn,"CID*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   if (GetStringKey(ppd->c_last,pIn,"CLT*",pTPCC,NAME_LEN))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   if (ppd->c_id == 0 && ppd->c_last[0] == 0)
   {
      strcpy(pTPCC->ErrTxt,"Error - Customer Id and Name Empty");
      pTPCC->iStatusId = ERR_IDANDNAME_EMPTY;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   if (ppd->c_id != 0 && ppd->c_last[0] != 0)
   {
      strcpy(pTPCC->ErrTxt,
         "Error - Specify Customer Id or Name, not Both");
```

```
   pTPCC->iStatusId = ERR_IDANDNAME_ENTERED;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
// Get and validate customer DId
if (GetShortKey(&ppd->c_d_id,pIn,"CDI*",pTPCC))
{
  FormatMenu(pOut,pTPCC);
  return(TRUE);
};
if (ppd->c_d_id < MIN_DId || ppd->c_d_id > MAX_DId)
{
   sprintf(pTPCC->ErrTxt,"Cust DId Out of Range(%ld,%ld) - %ld",
      MIN_DId,MAX_DId,ppd->d_id);
   pTPCC->iStatusId = ERR_DID_INVALID;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
// Get and validate customer WId
if (GetShortKey(&ppd->c_w_id,pIn,"CWI*",pTPCC))
{
  FormatMenu(pOut,pTPCC);
  return(TRUE);
};
if (ppd->c_w_id < 1)
{
   sprintf(pTPCC->ErrTxt,
      "Payment Contains Invalid Customer WId %d",
      ppd->c_w_id);
   pTPCC->iStatusId = ERR_WID_INVALID;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
// Get and validate amount
if (GetAmountKey(&ppd->h_amount,pIn,"HAM*",pTPCC))
{
  FormatMenu(pOut,pTPCC);
  return(TRUE);
};
if (ppd->h_amount <= 0)
{
   sprintf(pTPCC->ErrTxt,
      "Payment Amount Negative or Missing");
   pTPCC->iStatusId = ERR_AMOUNT_INVALID;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
bTMRslt = TMTran(PAYMENT_SERVICE,pTMon,&bTPRslt,&iTPRslt);
ppd = (PAYMENT_DATA *) pTMon->pTMData;
if (bTMRslt)
{
   pTPCC->iStatusId = ERR_TM_INTERFACE;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
if (bTPRslt)
{
   sprintf(pTPCC->ErrTxt,
      "Payment Service Returned Error(%ld): %s",
      iTPRslt,ppd->execution_status);
   pTPCC->iStatusId = ERR_SERVICE_RSLT;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
FormatRespHdr(pOut,"TPC-C Payment",pTPCC);
sprintf(pOut + strlen(pOut),
        "<PRE>                                  Payment<BR>"
   "Date: %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d <BR><BR>"
   "Warehouse: %4.4d"
   "                        District: %2.2d<BR>",
        ppd->h_date.day,ppd->h_date.month,
   ppd->h_date.year,ppd->h_date.hour,
   ppd->h_date.minute,ppd->h_date.second,
   ppd->w_id,ppd->d_id);

FormatHTMLString(szWork2,ppd->w_street_1,ADDR_LEN);
FormatHTMLString(szWork3,ppd->d_street_1,ADDR_LEN);
sprintf(pOut + strlen(pOut),
   "%s                      %s<BR>",szWork2,szWork3);
FormatHTMLString(szWork2,ppd->w_street_2,ADDR_LEN);
FormatHTMLString(szWork3,ppd->d_street_2,ADDR_LEN);
sprintf(pOut + strlen(pOut),
   "%s                      %s<BR>",szWork2,szWork3);
FormatHTMLString(pTPCC->szWork,ppd->w_city,ADDR_LEN);
FormatHTMLString(szWork2,ppd->d_city,ADDR_LEN);
FormatHTMLString(szWork3,ppd->w_state,STATE_LEN);
FormatHTMLString(szWork4,ppd->d_state,STATE_LEN);
FormatString(szZip1,ZIPPIC,ppd->w_zip);
FormatString(szZip2,ZIPPIC,ppd->d_zip);
sprintf(pOut +  strlen(pOut),
   "%s %s %10.10s       %s %s %10.10s<BR><BR>",
   pTPCC->szWork,szWork3,szZip1,szWork2,szWork4,szZip2);
FormatHTMLString(szWork2,ppd->c_first,NAME_LEN);
FormatHTMLString(szWork3,ppd->c_middle,2);
FormatHTMLString(szWork4,ppd->c_last,NAME_LEN);
sprintf(pOut + strlen(pOut),
   "Customer: %4.4d  Cust-Warehouse: %4.4d  Cust-District: %2.2d<BR>"
   "Name:   %s %s %s     Since:  %2.2d-%2.2d-%4.4d<BR>",
   ppd->c_id,ppd->c_w_id,ppd->c_d_id,
   szWork2,szWork3,szWork4,
   ppd->c_since.day,ppd->c_since.month,ppd->c_since.year);
FormatHTMLString(pTPCC->szWork,ppd->c_street_1,ADDR_LEN);
FormatHTMLString(szWork2,ppd->c_credit,2);
FormatHTMLString(szWork3,ppd->d_street_2,ADDR_LEN);
sprintf(pOut + strlen(pOut),
   "        %s                      Credit: %s<BR>"
   "        %s                      %%Disc:  %5.2f<BR>",
   pTPCC->szWork,szWork2,szWork3,ppd->c_discount * 100);
FormatHTMLString(szWork2,ppd->c_city,ADDR_LEN);
FormatHTMLString(szWork3,ppd->c_state,STATE_LEN);
FormatString(szZip1,ZIPPIC,ppd->c_zip);
FormatString(szWork4,"XXXXXX-XXX-XXX-XXXX",ppd->c_phone);
sprintf(pOut + strlen(pOut),
   "        %s %s %10.10s       Phone:  %-19.19s<BR><BR>"
   "Amount Paid:          $%7.2f      New Cust Balance: $%14.2f<BR>"
   "Credit Limit:   $%13.2f<BR><BR>",
   szWork2,szWork3,szZip1,szWork4,
   ppd->h_amount,ppd->c_balance,ppd->c_credit_lim);
pCredit = ppd->c_credit;
if (*pCredit == 'B' && *(pCredit + 1) == 'C')
{
   pCredit = ppd->c_data;
```

```
      iCDLines = strlen(pCredit) / 50;
      for(i = 0; i < 4; i++, pCredit += 50)
      {
         if (i <= iCDLines)
            UtilStrCpy(szWork2,pCredit,50);
         else
            szWork2[0] = 0;
         FormatHTMLString(szWork3,szWork2,50);
         if (!i)
            sprintf(pOut + strlen(pOut),
               "Cust-Data: %s<BR>",szWork3);
         else
            sprintf(pOut + strlen(pOut),
               "           %s<BR>",szWork3);
      };
   }
   else
      strcat(pOut,"Cust-Data: <BR><BR><BR><BR>");
   sprintf(pOut + strlen(pOut),
      "</PRE><HR><BR>%s</FORM>%s",szMenuList,HTMLTrailer);

   return(FALSE);

};   // ProcessPayment

//========================================================================
//
// Function name: ProcessDelivery
//
//    ProcessDelivery extracts the input data fields from pIn, processes
//    the data, and returns a response in pOut.
//
//    Result:
//       FALSE - Delivery processed successfully.
//       TRUE - Delivery processing failed.
//
//========================================================================
BOOL ProcessDelivery(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   DELIVERY_DATA * pdd;
   TMON_STATE * pTMon;
   BOOL bTMRslt;

   pTMon = &pTPCC->tsTMon;
   pTMon->lTMDataLen = sizeof(DELIVERY_DATA);
   memset(pTMon->pTMData,0,pTMon->lTMDataLen);
   pdd = (DELIVERY_DATA *) pTMon->pTMData;
   pdd->w_id = pTPCC->sWId;
   // Get and validate carrier id
   if (GetShortKey(&pdd->o_carrier_id,pIn,"OCD*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   if (pdd->o_carrier_id < MIN_CARRIER ||
       pdd->o_carrier_id > MAX_CARRIER)
   {
      sprintf(pTPCC->ErrTxt,"Carrier Id Out of Range(%ld,%ld) - %ld",
         MIN_CARRIER,MAX_CARRIER,pdd->o_carrier_id);
      pTPCC->iStatusId = ERR_CARRIER_INVALID;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   GetLocalTime(&pdd->QTime);
   bTMRslt = TMPost(DELIVERY_SERVICE,pTMon);
   if (bTMRslt)
   {
      pTPCC->iStatusId = ERR_TM_INTERFACE;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   strcpy(pdd->execution_status,"Delivery has been queued.");
   FormatRespHdr(pOut,"TPC-C Delivery",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                  Delivery<BR>"
      "Warehouse: %4.4d<BR><BR>"
      "Carrier Number: %2.2d<BR><BR>"
      "Execution Status: %25.25s<BR>",
      pdd->w_id,pdd->o_carrier_id,pdd->execution_status);
   sprintf(pOut + strlen(pOut),
      "</PRE><HR><BR>%s</FORM>%s",szMenuList,HTMLTrailer);

   return(FALSE);

};   // ProcessDelivery

//========================================================================
//
// Function name: ProcessOrderStatus
//
//    ProcessOrderStatus extracts the input data fields from pIn,
//    processes the data, and returns a response in pOut.
//
//    Result:
//       FALSE - OrderStatus processed successfully.
//       TRUE - OrderStatus processing failed.
//
//========================================================================
BOOL ProcessOrderStatus(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   ORDER_STATUS_DATA * posd;
   TMON_STATE * pTMon;
   INT i;
   CHAR szWork2[50];
   CHAR szWork3[50];
   BOOL bTMRslt;
   BOOL bTPRslt;
   INT iTPRslt;

   pTMon = &pTPCC->tsTMon;
   pTMon->lTMDataLen = sizeof(ORDER_STATUS_DATA);
   memset(pTMon->pTMData,0,pTMon->lTMDataLen);
   posd = (ORDER_STATUS_DATA *) pTMon->pTMData;
   posd->w_id = pTPCC->sWId;
   if (GetShortKey(&posd->d_id,pIn,"DID*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
   if (posd->d_id < MIN_DId || posd->d_id > MAX_DId)
```

```
{
   sprintf(pTPCC->ErrTxt,"DId Out of Range(%ld,%ld) - %ld",
      MIN_DId,MAX_DId,posd->d_id);
   pTPCC->iStatusId = ERR_DID_INVALID;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
if (GetLongKey(&posd->c_id,pIn,"CID*",pTPCC))
{
  FormatMenu(pOut,pTPCC);
  return(TRUE);
};
if (GetStringKey(posd->c_last,pIn,"CLT*",pTPCC,NAME_LEN))
{
  FormatMenu(pOut,pTPCC);
  return(TRUE);
};
if (posd->c_id == 0 && posd->c_last[0] == 0)
{
   strcpy(pTPCC->ErrTxt,"Error - Customer Id and Name Empty");
   pTPCC->iStatusId = ERR_IDANDNAME_EMPTY;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
if (posd->c_id != 0 && posd->c_last[0] != 0)
{
   strcpy(pTPCC->ErrTxt,
      "Error - Specify Customer Id or Name, not Both");
   pTPCC->iStatusId = ERR_IDANDNAME_ENTERED;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
bTMRslt = TMTran(ORDERSTATUS_SERVICE,pTMon,&bTPRslt,&iTPRslt);
posd = (ORDER_STATUS_DATA *) pTMon->pTMData;
if (bTMRslt)
{
   pTPCC->iStatusId = ERR_TM_INTERFACE;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
if (bTPRslt)
{
   sprintf(pTPCC->ErrTxt,
      "Order Status Service Returned Error(%ld): %s",
      iTPRslt,posd->execution_status);
   pTPCC->iStatusId = ERR_SERVICE_RSLT;
   FormatMenu(pOut,pTPCC);
   return(TRUE);
};
FormatRespHdr(pOut,"TPC-C Order-Status",pTPCC);
sprintf(pOut + strlen(pOut),
   "<PRE>                                Order-Status<BR>"
   "Warehouse: %4.4d   District: %2.2d<BR>",
   posd->w_id,posd->d_id);
FormatHTMLString(pTPCC->szWork,posd->c_first,NAME_LEN);
FormatHTMLString(szWork2,posd->c_middle,2);
FormatHTMLString(szWork3,posd->c_last,NAME_LEN);
sprintf(pOut + strlen(pOut),
   "Customer: %4.4d   Name: %s %s %s<BR>"
   "Cust-Balance: $%9.2f<BR><BR>",
   posd->c_id,pTPCC->szWork,szWork2,szWork3,posd->c_balance);
   sprintf(pOut + strlen(pOut),
      "Order-Number: %8.8d   Entry-Date: %2.2d-%2.2d-%4.4d
%2.2d:%2.2d:%2.2d   Carrier-Number: %2.2d<BR>"
      "Supply-W     Item-Id    Qty     Amount      Delivery-Date<BR>",
      posd->o_id,posd->o_entry_d.day,posd->o_entry_d.month,
      posd->o_entry_d.year,posd->o_entry_d.hour,
      posd->o_entry_d.minute,posd->o_entry_d.second,
      posd->o_carrier_id);
   for(i = 0; i < posd->o_ol_cnt; i++)
   {
      sprintf(pOut + strlen(pOut),
         "  %4.4d       %6.6d     %2.2d     $%8.2f      %2.2d-%2.2d-
%4.4d<BR>",
         posd->OlOrderStatusData[i].ol_supply_w_id,
         posd->OlOrderStatusData[i].ol_i_id,
         posd->OlOrderStatusData[i].ol_quantity,
         posd->OlOrderStatusData[i].ol_amount,
         posd->OlOrderStatusData[i].ol_delivery_d.day,
         posd->OlOrderStatusData[i].ol_delivery_d.month,
         posd->OlOrderStatusData[i].ol_delivery_d.year);
   };
   sprintf(pOut + strlen(pOut),
      "<BR></PRE><HR><BR>%s</FORM>%s",szMenuList,HTMLTrailer);

   return(FALSE);

};   // ProcessOrderStatus

//============================================================================
//
// Function name: ProcessStockLevel
//
//    ProcessStockLevel extracts the input data fields from pIn,
//    processes the data, and returns a response in pOut.
//
//    Result:
//       FALSE - StockLevel processed successfully.
//       TRUE - StockLevel processing failed.
//
//============================================================================
BOOL ProcessStockLevel(CHAR * pIn,CHAR * pOut,TPCC_STATE * pTPCC)
{
   STOCK_LEVEL_DATA * psld;
   TMON_STATE * pTMon;
   BOOL bTMRslt;
   BOOL bTPRslt;
   INT iTPRslt;

   pTMon = &pTPCC->tsTMon;
   pTMon->lTMDataLen = sizeof(STOCK_LEVEL_DATA);
   memset(pTMon->pTMData,0,pTMon->lTMDataLen);
   psld = (STOCK_LEVEL_DATA *) pTMon->pTMData;
   psld->w_id = pTPCC->sWId;
   psld->d_id = pTPCC->sDId;
   psld->low_stock = 0;
   psld->execution_status[0] = 0;
   if (GetShortKey(&psld->thresh_hold,pIn,"TT*",pTPCC))
   {
     FormatMenu(pOut,pTPCC);
     return(TRUE);
   };
```

```
   if (psld->thresh_hold < MIN_THRESHOLD ||
       psld->thresh_hold > MAX_THRESHOLD)
   {
      sprintf(pTPCC->ErrTxt,"Threshold Out of Range(%ld,%ld) - %ld",
         MIN_THRESHOLD,MAX_THRESHOLD,psld->thresh_hold);
      pTPCC->iStatusId = ERR_THRESHOLD_RANGE;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };

   bTMRslt = TMTran(STOCKLEVEL_SERVICE,pTMon,&bTPRslt,&iTPRslt);
   psld = (STOCK_LEVEL_DATA *) pTMon->pTMData;
   if (bTMRslt)
   {
      pTPCC->iStatusId = ERR_TM_INTERFACE;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };
   if (bTPRslt)
   {
      sprintf(pTPCC->ErrTxt,
         "Stock Level Service Returned Error(%ld): %s",
         iTPRslt,psld->execution_status);
      pTPCC->iStatusId = ERR_SERVICE_RSLT;
      FormatMenu(pOut,pTPCC);
      return(TRUE);
   };

   FormatRespHdr(pOut,"TPC-C Stock Level",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                Stock-Level<BR>"
      "Warehouse: %4.4d   District: %2.2d<BR><BR>"
      "Stock Level Threshold: %2.2d<BR><BR>"
      "low stock: %3.3ld</PRE><BR><HR>"
      "%s</FORM>%s",
      pTPCC->sWId,pTPCC->sDId,psld->thresh_hold,psld->low_stock,
      szMenuList,HTMLTrailer);

   return(FALSE);

};   // ProcessStockLevel

//==========================================================================
//
// Function name: GetHidden
//
//==========================================================================
BOOL GetHidden(CHAR * pMsg,UINT * uFormId,INT * iSyncId,INT * iTermId)
{
   CHAR * pPtr;
   BOOL bRslt = TRUE;

   // Extract TERMID
   pPtr = strstr(pMsg,TERMIDTOKEN);
   if (pPtr == NULL)
      goto xit;
   pPtr += strlen(TERMIDTOKEN);
   *iTermId = atoi(pPtr);

   // Extract SYNCID
   pPtr = strstr(pMsg,SYNCIDTOKEN);
   if (pPtr == NULL)
      goto xit;
   pPtr += strlen(SYNCIDTOKEN);
   *iSyncId = atoi(pPtr);

   // Extract FORMID
   pPtr = strstr(pMsg,FORMIDTOKEN);
   if (pPtr == NULL)
      goto xit;
   pPtr += strlen(FORMIDTOKEN);
   *uFormId = abs(atoi(pPtr));

   bRslt = FALSE;

xit:

   return(bRslt);

};   // GetHidden

//==========================================================================
//
// Function name: GetCmd
//
//==========================================================================
BOOL GetCmd(CHAR * pMsg,CHAR * pWork,UINT uLen)
{
   UINT u;
   CHAR * ptr;
   CHAR * pUpd;

   // Check for CMD key
   if (!(ptr = strstr(pMsg,CMDTOKEN)))
      return(CMD_NULL);
   ptr += sizeof(CMDTOKEN);
   pUpd = pWork;
   while (*ptr && *ptr != '&')
      *pUpd++ = *ptr++;
   *pUpd = 0;

   // Convert command name into command index
   for(u=0; u < CMD_MAX; u++)
   {
      if (!strcmp(szCmds[u],pWork))
         return(u);
   };

   // Command string not found
   return(CMD_NULL);

};   // GetCmd

//==========================================================================
//
// Function name: GetLongKey
//
//==========================================================================
BOOL GetLongKey(LONG * lRslt,CHAR * pHTML,CHAR * pKey,TPCC_STATE * pTPCC)
{
```

```
   if (GetKeyValue(pHTML,pKey,pTPCC->szWork,sizeof(pTPCC->szWork)))
   {
      sprintf(pTPCC->ErrTxt,"Error - Missing %s Key",pKey);
      pTPCC->iStatusId = ERR_MISSING_KEY;
      return(TRUE);
   };
   if (pTPCC->szWork[0] != 0 )
   {
      if (CheckNumeric(pTPCC->szWork))
      {
         sprintf(pTPCC->ErrTxt,"Error - %s Value Not Numeric",pKey);
         pTPCC->iStatusId = ERR_NOT_NUMERIC;
         return(TRUE);
      };
   };
   *lRslt = atol(pTPCC->szWork);
   return(FALSE);
};   // GetLongKey

//========================================================================
//
// Function name: GetIntKey
//
//========================================================================
BOOL GetIntKey(INT * iRslt,CHAR * pHTML,CHAR * pKey,TPCC_STATE * pTPCC)
{
   if (GetKeyValue(pHTML,pKey,pTPCC->szWork,sizeof(pTPCC->szWork)))
   {
      sprintf(pTPCC->ErrTxt,"Error - Missing %s Key",pKey);
      pTPCC->iStatusId = ERR_MISSING_KEY;
      return(TRUE);
   };
   if (pTPCC->szWork[0] != 0 )
   {
      if (CheckNumeric(pTPCC->szWork))
      {
         sprintf(pTPCC->ErrTxt,"Error - %s Value Not Numeric",pKey);
         pTPCC->iStatusId = ERR_NOT_NUMERIC;
         return(TRUE);
      };
   };
   *iRslt = atoi(pTPCC->szWork);
   return(FALSE);
};   // GetIntKey

//========================================================================
//
// Function name: GetShortKey
//
//========================================================================
BOOL GetShortKey(SHORT * sRslt,CHAR * pHTML,CHAR * pKey,TPCC_STATE *
pTPCC)
{
   if (GetKeyValue(pHTML,pKey,pTPCC->szWork,sizeof(pTPCC->szWork)))
   {
      sprintf(pTPCC->ErrTxt,"Error - Missing %s Key",pKey);
      pTPCC->iStatusId = ERR_MISSING_KEY;
      return(TRUE);
   };
   if (pTPCC->szWork[0] != 0 )
   {
      if (CheckNumeric(pTPCC->szWork))
      {
         sprintf(pTPCC->ErrTxt,"Error - %s Value Not Numeric",pKey);
         pTPCC->iStatusId = ERR_NOT_NUMERIC;
         return(TRUE);
      };
   };
   *sRslt = (SHORT) atoi(pTPCC->szWork);
   return(FALSE);
};   // GetShortKey

//========================================================================
//
// Function name: GetStringKey
//
//========================================================================
BOOL GetStringKey(CHAR * szRslt,CHAR * pHTML,CHAR * pKey,
                  TPCC_STATE * pTPCC,UINT uMax)
{
   UINT uLen;
   if (GetKeyValue(pHTML,pKey,pTPCC->szWork,sizeof(pTPCC->szWork)))
   {
      sprintf(pTPCC->ErrTxt,"Error - Missing %s Key",pKey);
      pTPCC->iStatusId = ERR_MISSING_KEY;
      return(TRUE);
   };
   uLen = strlen(pTPCC->szWork);
   if (uLen > uMax)
   {
      sprintf(pTPCC->ErrTxt,
         "Error - %s Key Input (%ld) Too Long (%ld)"
         ,pKey,uLen,uMax);
      pTPCC->iStatusId = ERR_INPUT_TOOLONG;
      return(TRUE);
   };
   _strupr(pTPCC->szWork);
   strcpy(szRslt,pTPCC->szWork);
   return(FALSE);
};   // GetStringKey

//========================================================================
//
// Function name: GetAmountKey
//
//========================================================================
BOOL GetAmountKey(DOUBLE * dRslt,CHAR * pHTML,CHAR * pKey,
                  TPCC_STATE * pTPCC)
{
   CHAR * ptr;
   BOOL bInvalid = FALSE;

   if (GetKeyValue(pHTML,pKey,pTPCC->szWork,sizeof(pTPCC->szWork)))
   {
      sprintf(pTPCC->ErrTxt,"Error - Missing %s Key",pKey);
      pTPCC->iStatusId = ERR_MISSING_KEY;
      return(TRUE);
   };
   ptr = pTPCC->szWork;
   while(*ptr)
   {
      if (*ptr == '.')
```

```
      {
         ptr++;
         if (!*ptr)
            break;
         if (*ptr < '0' || *ptr > '9')
         {
            bInvalid = TRUE;
            break;
         };
         ptr++;
         if (!*ptr)
            break;
         if (*ptr < '0' || *ptr > '9')
         {
            bInvalid = TRUE;
            break;
         };
         ptr++;
         if (*ptr)
         {
            bInvalid = TRUE;
            break;
         };
         break;
      }
      else
      if (*ptr < '0' || *ptr > '9')
      {
         bInvalid = TRUE;
         break;
      };
               ptr++;
       };   // while(!*ptr)

   if (!bInvalid)
      *dRslt = atof(pTPCC->szWork);
   else
   {
      sprintf(pTPCC->ErrTxt,
         "Error - Invalid Amount Format (%s)",pTPCC->szWork);
      pTPCC->iStatusId = ERR_AMOUNT_BADFORM;
   };

   return(bInvalid);

};   // GetAmountKey

//================================================================================
//
// Function name: GetKeyValue
//   This function parses an HTTP formatted string for specific key
//   values.  HTTP keys terminate with '='.  HTTP values terminate
//   with an '&' or '\0'.
//
//    Result:
//       FALSE - Key found, string value return in pValue
//       TRUE - Key not found
//
//================================================================================
BOOL GetKeyValue(CHAR * pHTML,CHAR * pKey,CHAR * pValue,UINT uMax)
{
   CHAR * ptr;
   if (!(ptr=strstr(pHTML,pKey)))
      return(TRUE);
   if (!(ptr=strchr(ptr,'=')))
      return(TRUE);
   ptr++;
   uMax--;
   while (*ptr && *ptr != '&' && uMax)
   {
      *pValue++ = *ptr++;
      uMax--;
   };
   *pValue = 0;
   return(FALSE);
};   // GetKeyValue

//================================================================================
//
// Function name: FormatLogin
//
//================================================================================
VOID FormatLogin(CHAR * pOut,CHAR * pAddText)
{
   sprintf(pOut,"%s<BR>%s<BR>%s",szFormLogin,pAddText,HTMLTrailer);
};   // FormatLogin

//================================================================================
//
// Function name: FormatMenu
//
//================================================================================
VOID FormatMenu(CHAR * pOut,TPCC_STATE * pTPCC)
{
   strcpy(pOut,"<HTML><HEAD><TITLE>TPC-C MainMenu</TITLE></HEAD><BODY>"
      "Select Desired Transaction.<BR><HR>"
      "<FORM ACTION=\"tpcc.dll\"METHOD=\"GET\">");
   sprintf(pOut + strlen(pOut),
      "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
      pTPCC->iStatusId,pTPCC->iTermId,pTPCC->iSyncId,FORM_MENU);
   sprintf(pOut + strlen(pOut),"%s</FORM><BR>%s<BR>%s",
      szMenuList,pTPCC->ErrTxt,HTMLTrailer);
};   // FormatMenu

//================================================================================
//
// Function name: FormatNewOrder
//
//================================================================================
VOID FormatNewOrder(CHAR * pOut,TPCC_STATE * pTPCC)
{
   pTPCC->uFormId = FORM_NEWORDER;
   FormatFormHdr(pOut,"TPC-C New Order",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                  New Order<BR>"
      "Warehouse: %4.4d   District: <INPUT NAME=\"DID*\" SIZE=1>
Date:<BR>"
```

```
              "Customer:  <INPUT NAME=\"CID*\" SIZE=4>   Name:
Credit:      %Disc:<BR>"
      "Order Number:           Number of Lines:           W_tax:
D_tax:<BR><BR>"
      " Supp_W  Item_Id  Item Name                Qty  Stock  B/G  Price
Amount<BR>"
      " <INPUT NAME=\"SP00*\" SIZE=4>  <INPUT NAME=\"IID00*\" SIZE=6>
<INPUT NAME=\"Qty00*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP01*\" SIZE=4>  <INPUT NAME=\"IID01*\" SIZE=6>
<INPUT NAME=\"Qty01*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP02*\" SIZE=4>  <INPUT NAME=\"IID02*\" SIZE=6>
<INPUT NAME=\"Qty02*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP03*\" SIZE=4>  <INPUT NAME=\"IID03*\" SIZE=6>
<INPUT NAME=\"Qty03*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP04*\" SIZE=4>  <INPUT NAME=\"IID04*\" SIZE=6>
<INPUT NAME=\"Qty04*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP05*\" SIZE=4>  <INPUT NAME=\"IID05*\" SIZE=6>
<INPUT NAME=\"Qty05*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP06*\" SIZE=4>  <INPUT NAME=\"IID06*\" SIZE=6>
<INPUT NAME=\"Qty06*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP07*\" SIZE=4>  <INPUT NAME=\"IID07*\" SIZE=6>
<INPUT NAME=\"Qty07*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP08*\" SIZE=4>  <INPUT NAME=\"IID08*\" SIZE=6>
<INPUT NAME=\"Qty08*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP09*\" SIZE=4>  <INPUT NAME=\"IID09*\" SIZE=6>
<INPUT NAME=\"Qty09*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP10*\" SIZE=4>  <INPUT NAME=\"IID10*\" SIZE=6>
<INPUT NAME=\"Qty10*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP11*\" SIZE=4>  <INPUT NAME=\"IID11*\" SIZE=6>
<INPUT NAME=\"Qty11*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP12*\" SIZE=4>  <INPUT NAME=\"IID12*\" SIZE=6>
<INPUT NAME=\"Qty12*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP13*\" SIZE=4>  <INPUT NAME=\"IID13*\" SIZE=6>
<INPUT NAME=\"Qty13*\" SIZE=1><BR>"
      " <INPUT NAME=\"SP14*\" SIZE=4>  <INPUT NAME=\"IID14*\" SIZE=6>
<INPUT NAME=\"Qty14*\" SIZE=1><BR>"
      "Execution Status:
Total:<BR><HR>"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Process\">"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Menu\">"
      "</FORM>%s",
      pTPCC->sWId,HTMLTrailer);
};   // FormatNewOrder

//================================================================================
//
// Function name: FormatPayment
//
//================================================================================
VOID FormatPayment(CHAR * pOut,TPCC_STATE * pTPCC)
{
   pTPCC->uFormId = FORM_PAYMENT;
   FormatFormHdr(pOut,"TPC-C Payment",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                   Payment<BR>"
      "Date:<BR><BR>"
      "Warehouse: %4.4d"
      "                          District: <INPUT NAME=\"DID*\"
SIZE=1><BR><BR><BR><BR><BR>"
      "Customer: <INPUT NAME=\"CID*\" SIZE=4>"
      "Cust-Warehouse: <INPUT NAME=\"CWI*\" SIZE=4>  "
      "Cust-District: <INPUT NAME=\"CDI*\" SIZE=1><BR>"
      "Name:                        <INPUT NAME=\"CLT*\" SIZE=16>
Since:<BR>"
      "                                                    Credit:<BR>"
      "                                                    Disc:<BR>"
      "                                                    Phone:<BR><BR>"
      "Amount Paid:          $<INPUT NAME=\"HAM*\" SIZE=7>      New Cust
Balance:<BR>"
      "Credit Limit:<BR><BR>Cust-Data: <BR><BR><BR><BR></PRE><HR>"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Process\">"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Menu\">"
      "</FORM>%s",
      pTPCC->sWId,HTMLTrailer);
};   // FormatPayment

//================================================================================
//
// Function name: FormatDelivery
//
//================================================================================
VOID FormatDelivery(CHAR * pOut,TPCC_STATE * pTPCC)
{
   pTPCC->uFormId = FORM_DELIVERY;
   FormatFormHdr(pOut,"TPC-C Delivery",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                   Delivery<BR>"
      "Warehouse: %4.4d<BR><BR>"
      "Carrier Number: <INPUT NAME=\"OCD*\" SIZE=1><BR><BR>"
      "Execution Status:<BR></PRE><HR>"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Process\">"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Menu\">"
      "</FORM>%s",
      pTPCC->sWId,HTMLTrailer);
};   // FormatDelivery

//================================================================================
//
// Function name: FormatOrderStatus
//
//================================================================================
VOID FormatOrderStatus(CHAR * pOut,TPCC_STATE * pTPCC)
{
   pTPCC->uFormId = FORM_ORDERSTATUS;
   FormatFormHdr(pOut,"TPC-C Order-Status",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                 Order-Status<BR>"
      "Warehouse: %4.4d   "
      "District: <INPUT NAME=\"DID*\" SIZE=1><BR>"
               "Customer: <INPUT NAME=\"CID*\" SIZE=4>   Name:
<INPUT NAME=\"CLT*\" SIZE=23><BR>"
      "Cust-Balance:<BR><BR>"
      "Order-Number:            Entry-Date:                        Carrier-
Number:<BR>"
      "Supply-W     Item-Id    Qty     Amount      Delivery-
Date<BR></PRE><HR>"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Process\">"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Menu\">"
      "</FORM>%s",
      pTPCC->sWId,HTMLTrailer);
};   // FormatOrderStatus
```

```
//==============================================================================
//
// Function name: FormatStockLevel
//
//==============================================================================
VOID FormatStockLevel(CHAR * pOut,TPCC_STATE * pTPCC)
{
   pTPCC->uFormId = FORM_STOCKLEVEL;
   FormatFormHdr(pOut,"TPC-C Stock Level",pTPCC);
   sprintf(pOut + strlen(pOut),
      "<PRE>                                Stock-Level<BR>"
      "Warehouse: %4.4d   District: %2.2d<BR><BR>"
      "Stock Level Threshold: <INPUT NAME=\"TT*\"SIZE=2><BR><BR>"
      "low stock:    <BR><HR>"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Process\">"
      "<INPUT TYPE=\"submit\"NAME=\"CMD\" VALUE=\"Menu\">"
      "</FORM>%s",
      pTPCC->sWId,pTPCC->sDId,HTMLTrailer);
};   // FormatStockLevel

//==============================================================================
//
// Function name: FormatFormHdr
//
//==============================================================================
VOID FormatFormHdr(CHAR * pOut,CHAR * pTitle,TPCC_STATE * pTPCC)
{
   sprintf(pOut,
      "<HTML><HEAD><TITLE>%s</TITLE></HEAD>"
      "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">"
      "<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">"
      "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">"
      "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">",
      pTitle,pTPCC->uFormId,pTPCC->iTermId,pTPCC->iSyncId);
};   // FormatFormHdr

//==============================================================================
//
// Function name: FormatRespHdr
//
//==============================================================================
VOID FormatRespHdr(CHAR * pOut,CHAR * pTitle,TPCC_STATE * pTPCC)
{
   sprintf(pOut,
      "<HTML><HEAD><TITLE>%s</TITLE></HEAD>"
      "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">"
      "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">"
      "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">",
      pTitle,pTPCC->iStatusId,pTPCC->uFormId,
      pTPCC->iTermId,pTPCC->iSyncId);
};   // FormatRespHdr

//==============================================================================
//
// Function name: FormatHTMLString
//
//    Encodes HTML special characters.  If necessary, space fills
//    to pOut to total uLen characters.
//
//==============================================================================
VOID FormatHTMLString(CHAR * pOut,CHAR * pIn,UINT uLen)
{
   while (uLen && *pIn)
   {
      switch (*pIn)
      {
         case '>':
            *pOut++ = '&';
            *pOut++ = 'g';
            *pOut++ = 't';
            *pOut++ = ';';
            pIn++;
            break;
         case '<':
            *pOut++ = '&';
            *pOut++ = 'l';
            *pOut++ = 't';
            *pOut++ = ';';
            pIn++;
            break;
         case '&':
            *pOut++ = '&';
            *pOut++ = 'a';
            *pOut++ = 'm';
            *pOut++ = 'p';
            *pOut++ = ';';
            pIn++;
            break;
         case '\"':
            *pOut++ = '&';
            *pOut++ = 'q';
            *pOut++ = 'u';
            *pOut++ = 'o';
            *pOut++ = 't';
            *pOut++ = ';';
            pIn++;
            break;
         default:
            *pOut++ = *pIn++;
            break;
      };   // switch (*pIn)
      uLen--;
   };   // while (uLen && *pIn)
   while(uLen--)
      *pOut++ = ' ';
       *pOut = 0;
};   // FormatHTMLString

//==============================================================================
//
// Function name: FormatString
//
//    Encodes formatted string for HTML transmission.
//
//==============================================================================
VOID FormatString(CHAR * pOut,CHAR * pPic,CHAR * pIn)
```

```
{
   while(*pPic)
   {
      if (*pPic == 'X' )
      {
         if (*pIn)
            *pOut++ = *pIn++;
         else
            *pOut++ = ' ';
      }
      else
         *pOut++ = *pPic;
      pPic++;
   };
   *pOut = 0;
};   // FormatString

//=========================================================================
// FUNCTION: UtilStrCpy
//
//    Copies n characters from string pSrc to pDst and places a null
//    null character at the end of the destination string.  Unlike
//    strncpy this function ensures that the result string is always
//    null terminated.
//
//=========================================================================
VOID UtilStrCpy(CHAR * pDest,CHAR * pSrc,INT n)
{
   strncpy(pDest,pSrc,n);
   pDest[n] = '\0';
   return;
};   // UtilStrCpy

//=========================================================================
//
// Function name: CheckNumeric
//
//    Result
//       FALSE - string is all numeric
//       TRUE - sting contains non-numeric characters
//
//=========================================================================
BOOL CheckNumeric(CHAR * pNum)
{
   if (*pNum == 0 )
      return(TRUE);
   while (*pNum && isdigit(*pNum))
      pNum++;
   return(*pNum);
};   // CheckNumeric
```

term.h

```
// term.h

#include <timeb.h>

#define TMILLI_TIMEOUT 3600000       // One hour

typedef struct
{
   BOOL bInUse;                        // In use flag
   INT iTermId;                        // TermId
   LPVOID ConnID;                      // Connection Id
   INT  iSyncId;                       // Sync Id
   SHORT sWId;                         // TPCC WareHouse Id
   SHORT sDId;                         // TPCC District Id
   struct _timeb tbLastAccess;         // Last activity timestamp
} TERM_STATE;

BOOL TermInit(INT iSetMaxTerm);
VOID TermTerm(VOID);
TERM_STATE * TermAlloc(VOID);
TERM_STATE * TermGet(INT iTermId);
BOOL TermFree(INT iTermId);
```

term.c

```
// term.c
//
// Copyright Unisys, 1997
//
#include <windows.h>
#include <stdio.h>
#include "diagio.h"
#include "timesupp.h"
#include "term.h"

TERM_STATE * pTArray;
INT iNextTerm = 0;
INT iMaxTerm = 0;
CRITICAL_SECTION csTerm;

VOID TermMaint(VOID);

//=========================================================================
//
// Function name: TermInit
//    Creates and initializes the first TERMINITIAL TArray entries.
//    Initializes critical section to control access to TArray.  Assumes
//    access to function is single threaded, no other threads will start
//    until this function completes and that function is called once
//    (DLL_PROCESS_ATTACH).
//
// Returns:
//    FALSE TArray allocated and initialized
//    TRUE  TArray allocation failure
//
//=========================================================================
BOOL TermInit(INT iSetMaxTerm)
{
   INT iTermId;
   CHAR szDiag[MAX_DIAG_SZ];
   if (pTArray != NULL)
   {
      sprintf(szDiag,"TermInit(%ld): TArray Already Initialized\n",
         GetCurrentThreadId());
      DiagIoWrite(szDiag,DIAG_ERROR);
      return(TRUE);
   };
```

```
   InitializeCriticalSection(&csTerm);
   iMaxTerm = iSetMaxTerm;
   pTArray = (TERM_STATE *) malloc(sizeof(TERM_STATE) * (iMaxTerm + 1));
   if (pTArray == NULL)
   {
      sprintf(szDiag,"TermInit(%ld): malloc failed (%ld)\n",
         GetCurrentThreadId(),GetLastError());
      DiagIoWrite(szDiag,DIAG_ERROR);
      return(TRUE);
   }
   for (iTermId = 1; iTermId <= iMaxTerm; iTermId++)
      TermFree(iTermId);
   iNextTerm = 1;
   return(FALSE);
};   // TermInit

//=========================================================================
//
// Function name: TermTerm
//    Frees TArray and deletes csTerm critical section.  Assumes access
//    to function is single threaded and no other threads are actively
//    accessing TArray entries (DLL_PROCESS_DETACH).
//
//=========================================================================
VOID TermTerm(VOID)
{
   DeleteCriticalSection(&csTerm);
   if (pTArray != NULL)
      free(pTArray);
   iNextTerm = 0;
   iMaxTerm =0;
};   // TermTerm

//=========================================================================
//
// Function name: TermAlloc
//    Allocates empty TArray.  Uses iNextTerm to start search.
//
// Returns:
//    > 0   TArray entry index (iTermId)
//    < 0   Empty TArray entry not available
//
//=========================================================================
TERM_STATE * TermAlloc(VOID)
{
   INT iTermId = -1;
   if (pTArray == NULL)
   {
      CHAR szDiag[MAX_DIAG_SZ];
      sprintf(szDiag,"TermAlloc(%ld): Term Array Not Allocated\n",
         GetCurrentThreadId());
      DiagIoWrite(szDiag,DIAG_ERROR);
      return(NULL);
   };
   EnterCriticalSection(&csTerm);
   __try
   {
      while(iNextTerm <= iMaxTerm)
      {
         if (!pTArray[iNextTerm].bInUse)
         {
            pTArray[iNextTerm].bInUse = TRUE;
            _ftime(&pTArray[iNextTerm].tbLastAccess);
            iTermId = iNextTerm;
            iNextTerm++;
            break;
         };
         iNextTerm++;
      };   // while(iNextTerm <= iMaxTerm) (1st Try)
      if (iTermId <= 0)
      {
         // No entry found.  Perform maint and try again
         TermMaint();
         iNextTerm = 1;
         while(iNextTerm <= iMaxTerm)
         {
            if (!pTArray[iNextTerm].bInUse)
            {
               pTArray[iNextTerm].bInUse = TRUE;
               _ftime(&pTArray[iNextTerm].tbLastAccess);
               iTermId = iNextTerm;
               iNextTerm++;
               break;
            };
            iNextTerm++;
         };   // while(iNextTerm <= iMaxTerm) (2nd Try)
      };   // if (iTermId <= 0)
      if (iTermId <= 0)
         iNextTerm = 1;
   }
   __finally
   {
      LeaveCriticalSection(&csTerm);
   };

   if (iTermId > 0)
      return(&pTArray[iTermId]);
   else
      return(NULL);


};   // TermAlloc

//=========================================================================
//
// Function name: TermMaint
//    Clears entries whose last access time exceeds TMILLI_TIMEOUT.
//    Assumes caller has entered csTerm.
//
//=========================================================================
VOID TermMaint(VOID)
{
   INT iTermId;
   TMILLI tmElapsed;
   // Free entries that have timed out
   for (iTermId = 1; iTermId <= iMaxTerm; iTermId++)
   {
      if (pTArray[iTermId].bInUse)
      {
         tmElapsed = TimebElapsed(&pTArray[iTermId].tbLastAccess);
```

```
         if (tmElapsed > TMILLI_TIMEOUT)
            TermFree(iTermId);
      };
   };
};   // TermMaint

//=============================================================================
//
// Function name: TermGet
//    Returns pointer to TArray slot at iTermId.
//
// Returns:
//    FALSE TArray entry made available
//    TRUE  iTermId invalid.
//
//=============================================================================
TERM_STATE * TermGet(INT iTermId)
{
   TERM_STATE * pTerm;
   TMILLI tmElapsed;
   if (iTermId <= 0 || iTermId > iMaxTerm)
   {
      CHAR szDiag[MAX_DIAG_SZ];
      sprintf(szDiag,"TermGet(%ld): Invalid TermId (%ld)\n",
         GetCurrentThreadId(),iTermId);
      DiagIoWrite(szDiag,DIAG_ERROR);
      return(NULL);
   };
   pTerm = &pTArray[iTermId];
   if (!pTerm->bInUse)
      return(NULL);
   tmElapsed = TimebElapsed(&pTerm->tbLastAccess);
   if (tmElapsed > TMILLI_TIMEOUT)
      return(NULL);   // Entry destined to be freed by maint
   _ftime(&pTArray[iTermId].tbLastAccess);
   return(&pTArray[iTermId]);
};   // TermGet

//=============================================================================
//
// Function name: TermFree
//    Initializes contents of TArray slot at iTermId.
//
// Returns:
//    FALSE TArray entry made available
//    TRUE  iTermId invalid.
//
//=============================================================================
BOOL TermFree(INT iTermId)
{
   TERM_STATE * pTerm;
   if (iTermId <= 0 || iTermId > iMaxTerm)
   {
      CHAR szDiag[MAX_DIAG_SZ];
      sprintf(szDiag,"TermFree(%ld): Invalid TermId (%ld)\n",
         GetCurrentThreadId(),iTermId);
      DiagIoWrite(szDiag,DIAG_ERROR);
      return(TRUE);
   };
   pTerm = &pTArray[iTermId];
   pTerm->ConnID = 0;
   pTerm->sWId = 0;
   pTerm->sDId = 0;
   pTerm->iSyncId = 0;
   pTerm->iTermId = iTermId;
   TimebClear(&pTerm->tbLastAccess);
   pTerm->bInUse = FALSE;
};   // TermFree
```

timesupp.h

```
// timesupp.h

#include <windows.h>
#include <time.h>
#include <timeb.h>

#define TIMEB_STRING_SZ 23
#define TIMEB_STRING_DATESZ 10
#define TIMEB_STRING_TIMEOFFSET 11
#define TIMEB_STRING_TIMESZ 12

typedef ULONG TMILLI;

TMILLI TimebDiff(struct _timeb * p_tb1, struct _timeb * p_tb2);
VOID TimebCopy(struct _timeb * p_tbDest, struct _timeb * p_tbSource);
TMILLI TimebElapsed(struct _timeb * p_tb1);
VOID TimebClear(struct _timeb * p_tb1);
CHAR * TimebToString(struct _timeb * p_tb1,CHAR * psz,BOOL bMillis);
BOOL TimebFromString(struct _timeb * p_tb1,CHAR * psz);
VOID TimebAddSecs(struct _timeb * p_tb1,INT iSeconds);
```

timesupp.c

```
// timesupp.c
//
// Copyright Unisys, 1997
//

#include <stdio.h>
#include "timesupp.h"

//=============================================================================
//
// Function name: TimebCopy
//    Structure contents copy of _timeb source to _timeb dest.
//
//=============================================================================
VOID TimebCopy(struct _timeb * p_tbDest, struct _timeb * p_tbSource)
{
   p_tbDest->time = p_tbSource->time;
   p_tbDest->millitm = p_tbSource->millitm;
   p_tbDest->dstflag = p_tbSource->dstflag;
   p_tbDest->timezone = p_tbSource->timezone;

};   // TimebCopy

//=============================================================================
//
// Function name: TimebDiff
```

```
//    Time difference in milliseconds between _timeb _t1 and _timeb _t2.
//
//=========================================================================
TMILLI TimebDiff(struct _timeb * p_tb1, struct _timeb * p_tb2)
{
   LONG lRslt;
   lRslt = ((p_tb2->time - p_tb1->time) * 1000) +
               (p_tb2->millitm - p_tb1->millitm);
   if (lRslt < 0)
      return(0);
   else
      return((TMILLI) lRslt);

};   // TimebDiff

//=========================================================================
//
// Function name: TimebElaspsed
//
//=========================================================================
TMILLI TimebElapsed(struct _timeb * p_tb1)
{
   struct _timeb _tb2;
   _ftime(&_tb2);
   return (TimebDiff(p_tb1,&_tb2));

};   // TimebElapsed

//=========================================================================
//
// Function name: TimebClear
//
//=========================================================================
VOID TimebClear(struct _timeb * p_tb1)
{
   p_tb1->time = 0;
   p_tb1->millitm = 0;

};   // TimebClear

//=========================================================================
//
// Function name: TimebToString
//    Converts timeb to yyyy:mm:dd,hh:mm:ss.sss format
//
//=========================================================================
CHAR * TimebToString(struct _timeb * p_tb1,CHAR * psz,BOOL bMillis)
{
   struct tm * ptm;
   ptm = localtime(&p_tb1->time);
   sprintf(psz,"%4.4d/%2.2d/%2.2d,%2.2d:%2.2d:%2.2d",
      ptm->tm_year + 1900,ptm->tm_mon + 1,ptm->tm_mday,
      ptm->tm_hour,ptm->tm_min,ptm->tm_sec);
   if (bMillis)
      sprintf(psz + strlen(psz),".%3.3d",p_tb1->millitm);
   return(psz);
};   // TimebToString

//=========================================================================
//
// Function name: TimebFromString
//    Converts yyyy:mm:dd,hh:mm:ss.sss (TimebToString) format to timeb
//
//=========================================================================
BOOL TimebFromString(struct _timeb * p_tb1,CHAR * psz)
{
   struct tm tmTime;
   struct tm * ptm;
   UINT uLen;

   ptm = &tmTime;
   uLen = strlen(psz);
   if (uLen < (TIMEB_STRING_SZ - 4))   // millis are optional
   {
      p_tb1->time = 0;
      p_tb1->millitm = 0;
      return (TRUE);
   };
   // Clear fields that won't be set
   ptm->tm_wday = 0;
   ptm->tm_yday = 0;
   ptm->tm_isdst = -1;
   // Set tm struct fields from string
   ptm->tm_year = (atoi(psz)) - 1900;
   psz += 5;
   ptm->tm_mon = (atoi(psz)) - 1;
   psz += 3;
   ptm->tm_mday = atoi(psz);
   psz += 3;
   ptm->tm_hour = atoi(psz);
   psz += 3;
   ptm->tm_min = atoi(psz);
   psz +=3;
   ptm->tm_sec = atoi(psz);
   if (uLen >= TIMEB_STRING_SZ)   // Millis present
   {
      psz += 3;
      p_tb1->millitm = atoi(psz);
   };
   p_tb1->time = mktime(ptm);
   return(FALSE);
};   // TimebFromString

//=========================================================================
//
// Function name: TimebAddSecs
//
//=========================================================================
VOID TimebAddSecs(struct _timeb * p_tb1,INT iSeconds)
{
   p_tb1->time += iSeconds;
};   // TimebAddSecs
```

diagio.h

```
// diagio.h

// Environment variable defaults
#define DEFAULTDIAGLEVEL DIAG_INFO
```

```
#define DEFAULTEVENTLOG 0

#define DIAGNOSTICS TRUE
#define MAX_DIAG_SZ 2000

// Severity level of diagnostic report
#define DIAG_FORCE 1
#define DIAG_ERROR 2
#define DIAG_STATE 3
#define DIAG_INFO 4

VOID DiagIoInit(CHAR * pDiagId,BOOL bConsole,BOOL bEvent,UINT uLevel);
VOID DiagIoTerm(VOID);
VOID DiagIoWrite(CHAR * pDiagBuffer, UINT uSeverity);
```

diagio.c

```
// diagio.c
//
// Copyright Unisys, 1997
//
#include <windows.h>
#include <stdio.h>
#include "diagio.h"

CRITICAL_SECTION csDiagIo;
HANDLE hEventLog = NULL;
UINT uDiagLevel;
BOOL bEventLog;
BOOL bConsoleLog;
CHAR * pDiagHdr;
CHAR * pEventHost;
CHAR * pErrHdr =
   {"*** ERROR *** ERROR *** ERROR *** ERROR *** ERROR ***"};

INT WriteEventLog(CHAR * pDMsgs[],UINT uMsgCnt,UINT uSeverity);

//============================================================================
//
// Function name: DiagIoInit
//
//============================================================================
VOID DiagIoInit(CHAR * pDiagId,BOOL bConsole,BOOL bEvent,UINT uLevel)
{
   if (DIAGNOSTICS)
   {
      InitializeCriticalSection(&csDiagIo);

      uDiagLevel = uLevel;
      bEventLog = bEvent;
      bConsoleLog = bConsole;
      pEventHost = (CHAR *) malloc(10);
      strcpy(pEventHost,"");  // local host
      pDiagHdr = (CHAR *) malloc(strlen(pDiagId) + 1);
      strcpy(pDiagHdr,pDiagId);
      if (bEventLog)
      {
         hEventLog = RegisterEventSource(pEventHost,pDiagId);
         if (hEventLog == NULL)
         {
            bEventLog = FALSE;
            if (bConsoleLog)
               fprintf(stdout,
                  "%s: Event Log Register Failed (%ld)\n"
                  "Event Log Will NOT be Used\n",
                  pDiagHdr,GetLastError());
         }
         else
         {
            if (bConsoleLog)
               fprintf(stdout,"%s: Event Logging to LocalHost as %s\n",
                  pDiagHdr,pDiagHdr);
         };
      };   // if bEventLog
   };   // if Diagnostics
};   // DiagIoInit


//============================================================================
//
// Function name: DiagIoTerm
//
//============================================================================
VOID DiagIoTerm(VOID)
{
   if (DIAGNOSTICS)
   {
      DeleteCriticalSection(&csDiagIo);
      if (hEventLog != NULL)
         DeregisterEventSource(hEventLog);
      free(pDiagHdr);
      free(pEventHost);
   };
};   // DiagIoTerm

//============================================================================
//
// Function name: DiagIoWrite
//
//============================================================================
VOID DiagIoWrite(CHAR * pDiagBuffer, UINT uSeverity)
{
   CHAR * pDMsgs[3];
   UINT uMsgCnt = 0;
   INT iERslt = 0;
   if (DIAGNOSTICS)
   {
      if (uDiagLevel >= uSeverity)
      {
         EnterCriticalSection(&csDiagIo);
         __try
         {
            if (uSeverity == DIAG_ERROR)
            {
               pDMsgs[0] = pDiagHdr;
               pDMsgs[1] = pErrHdr;
               pDMsgs[2] = pDiagBuffer;
               uMsgCnt = 3;
            }
            else
            {
```

```
               pDMsgs[0] = pDiagHdr;
               pDMsgs[1] = pDiagBuffer;
               uMsgCnt = 2;
            };
            if (bEventLog)
               iERslt = WriteEventLog(pDMsgs,uMsgCnt,uSeverity);
            if (bConsoleLog)
            {
               if (uMsgCnt == 3)
                  fprintf(stdout,"\n%s:
%s\n%s",pDMsgs[0],pDMsgs[1],pDMsgs[2]);
               else
                  fprintf(stdout,"\n%s: %s",pDMsgs[0],pDMsgs[1]);
               if (iERslt != 0)
                  fprintf(stdout,
                     "EventLog Write Failed (%ld), No Longer in Use\n",
                     iERslt);
            };
         }
         __finally
         {
            LeaveCriticalSection(&csDiagIo);
         };
      };   // if uDiagLevel >= uSeverity
   };   // if Diagnostics
};   // DiagIoWrite

INT WriteEventLog(CHAR * pDMsgs[],UINT uMsgCnt,UINT uSeverity)
{
   WORD wType;
   WORD wCount;
   wCount = uMsgCnt;
   switch (uSeverity)
   {
      case DIAG_ERROR:
         wType = EVENTLOG_ERROR_TYPE;
         break;
      default:
         wType = EVENTLOG_INFORMATION_TYPE;
         break;
   };
   if (wType != 0)
   {
      if (!ReportEvent(hEventLog,           // event log handle
                       wType,               // event type
                       0,                   // category zero
                       uSeverity,           // no event identifier
                       NULL,                // no user security identifier
                       wCount,              // # of substitution strings
                       0,                   // no binary data
                       (LPCTSTR *) pDMsgs,  // address of string array
                       NULL))               // address of binary
      {
         DeregisterEventSource(hEventLog);
         hEventLog = NULL;
         bEventLog = FALSE;
         return(GetLastError());
      };   // ReportEvent failed
   };   // if wType != 0
   return(0);
};   // WriteEventLog
```

SERVER MAKEFILES

```
SVR = tpccsvr
SRC = tpccsvr.c
DBG = /f "/Zi"
$(SVR).exe: $(SRC)
   erase $(SVR).exe
   $(TUXDIR)\bin\buildserver /f "$(SRC)" /o $(SVR).exe /s
NEWORDER:NEWORDER /s PAYMENT:PAYMENT /s ORDERSTS:ORDERSTS /s
STOCKLVL:STOCKLVL -l d:\mssql\dblib\lib\ntwdblib.lib
   copy $(SVR).exe $(APPDIR)

SVR = tpccdelv
SRC = tpccdelv.c
DBG = /f "/Zi"
$(SVR).exe: $(SRC)
   erase $(SVR).exe
   $(TUXDIR)\bin\buildserver /f "$(SRC)" /o $(SVR).exe /s
DELIVERY:DELIVERY  -l d:\mssql\dblib\lib\ntwdblib.lib
   copy $(SVR).exe $(APPDIR)
```

tpccsvr.h

```
// tpccsvr.h
//
// Copyright Unisys, 1997
// Copyright Microsoft, 1996

#include "tpcc.h"

#define DEFCLPACKSIZE            2000
#define DEADLOCKWAIT             10
#define LOGFILE_NAME             "delilog"

// String length constants
#define SERVER_NAME_LEN            20
#define DATABASE_NAME_LEN          20
#define USER_NAME_LEN              20
#define PASSWORD_LEN               20
#define TABLE_NAME_LEN             20
```

tpccsvr.c

```
// tpccsvr.c
//
// Copyright Unisys, 1997
// Copyright Microsoft, 1996

#include <windows.h>
#include <malloc.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
```

```
#include <atmi.h>
#include <userlog.h>

#include "tpccsvr.h"

char    szServer[32]   = "tpccserver";
char    szUser[32]             = { 0 };
char    szPassword[32] = { 0 };
char    szDatabase[32] = "tpcc";
char  szService[16]  = "tpccsvr";
char  szWork[200];
PDBPROCESS     dbproc;
int     spid;                                   // spid assigned from dblib
BOOL  bFailed;
BOOL  bDeadlock;
short   DeadlockRetry  = (short)3;

int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr,
char *dberrstr, char *oserrstr);
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext);
int SQLStockLevel(STOCK_LEVEL_DATA *psld);
int SQLNewOrder(NEW_ORDER_DATA * pnod);
int SQLPayment(PAYMENT_DATA *ppd);
int SQLOrderStatus(ORDER_STATUS_DATA * pOrderStatus);
void UtilStrCpy(char * pDest, char * pSrc, int n);
VOID GetArgs(INT argc, CHAR **argv);

//============================================================================
//
// Function name: tpsvrinit
//
//============================================================================
tpsvrinit(int argc, char *argv[])
{
   GetArgs(argc,argv);
   sprintf(szWork,"%s Started, DBServer=%s,DB=%s",
      szService,szServer,szDatabase);
   userlog(szWork);
   if (SQLInit(szServer,szDatabase,szUser,szPassword))
      return(-1);
   userlog("Database open, initialization complete");
   return(0);
};   // tpsvrinit

//============================================================================
//
// Function name: tpsvrdone
//
//============================================================================
void tpsvrdone()
{
   userlog("Shutdown request for tpcctux server");
   dbclose(dbproc);
   dbexit();
};   // tpsvrdone

//============================================================================
//
// Function name: NEWORDER
//
//    Entry point called by tuxedo for NEWORDER service requests.
//
//============================================================================
void NEWORDER(TPSVCINFO * svcinfo)
{
   int iRslt;
   NEW_ORDER_DATA * pnod;

   pnod = (NEW_ORDER_DATA *) svcinfo->data;
   iRslt = SQLNewOrder(pnod);

   // Check for DBLib termination error
   if (bFailed)
   {
      strcpy(pnod->execution_status,szWork);
      tpreturn(TPFAIL,SVCERR_DBLIB,svcinfo->data,svcinfo->len,0);
   }
   else
   if (iRslt == 0)
      tpreturn(TPSUCCESS,0,svcinfo->data,svcinfo->len,0);
   else
      tpreturn(TPFAIL,iRslt,svcinfo->data,svcinfo->len,0);
};   // NEWORDER

//============================================================================
//
// Function name: PAYMENT
//
//    Entry point called by tuxedo for PAYMENT service requests.
//
//============================================================================
void PAYMENT(TPSVCINFO * svcinfo)
{
   int iRslt;
   PAYMENT_DATA * ppd;

   ppd = (PAYMENT_DATA *) svcinfo->data;

   iRslt = SQLPayment(ppd);

   if (bFailed)
   {
      strcpy(ppd->execution_status,szWork);
      tpreturn(TPFAIL,SVCERR_DBLIB,svcinfo->data,svcinfo->len,0);
   }
   else
   if (iRslt == 0)
      tpreturn(TPSUCCESS,0,svcinfo->data,svcinfo->len,0);
   else
      tpreturn(TPFAIL,iRslt,svcinfo->data,svcinfo->len,0);
};   // PAYMENT

//============================================================================
//
// Function name: ORDERSTS
//
//    Entry point called by tuxedo for ORDERSTS service requests.
//
//============================================================================
void ORDERSTS(TPSVCINFO * svcinfo)
```

```
{
   int iRslt;
   ORDER_STATUS_DATA * posd;

   posd = (ORDER_STATUS_DATA *) svcinfo->data;
   iRslt = SQLOrderStatus(posd);

   // Check for DBLib termination error
   if (bFailed)
   {
      strcpy(posd->execution_status,szWork);
      tpreturn(TPFAIL,SVCERR_DBLIB,svcinfo->data,svcinfo->len,0);
   }
   else
   if (iRslt == 0)
      tpreturn(TPSUCCESS,0,svcinfo->data,svcinfo->len,0);
   else
      tpreturn(TPFAIL,iRslt,svcinfo->data,svcinfo->len,0);
};   // ORDERSTS

//============================================================================
//
// Function name: STOCKLVL
//
//    Entry point called by tuxedo for STOCKLVL service requests.
//
//============================================================================
void STOCKLVL(TPSVCINFO * svcinfo)
{
   int iRslt;
   STOCK_LEVEL_DATA * psld;

   psld = (STOCK_LEVEL_DATA *) svcinfo->data;
   iRslt = SQLStockLevel(psld);

   // Check for DBLib termination error
   if (bFailed)
   {
      strcpy(psld->execution_status,szWork);
      tpreturn(TPFAIL,SVCERR_DBLIB,svcinfo->data,svcinfo->len,0);
   }
   else
   if (iRslt == 0)
      tpreturn(TPSUCCESS,0,svcinfo->data,svcinfo->len,0);
   else
      tpreturn(TPFAIL,iRslt,svcinfo->data,svcinfo->len,0);

};   // STOCKLVL

//============================================================================
//
// Function name: SQLInit
//
//    Set global dbproc and spid.
//
//    Result:
//       FALSE - database open, dbproc valid
//       TRUE - database open failed
//
//============================================================================
BOOL SQLInit(CHAR * pSvr,CHAR * pDB,CHAR * pUsr,CHAR * pPW,CHAR * pSvc)
{
   char szApp[32];
   char server[256];
   char database[256];
   char user[256];
   char password[256];
   LOGINREC    *login;

   dbinit();
   // install error and message handlers
   dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
   dberrhandle((DBERRHANDLE_PROC)err_handler);

   dbproc = NULL;
   strcpy(server,pSvr);
   strcpy(database,pDB);
   strcpy(user,pUsr);
   strcpy(password,pPW);
   sprintf(szApp,"%s%ld",pSvc,_getpid());

   login = dblogin();
   if (!*user )
      DBSETLUSER(login,"sa");
   else
      DBSETLUSER(login,user);
   DBSETLPWD(login,password);
   DBSETLHOST(login,szApp);
// DBSETLPACKET(login,(unsigned short)DEFCLPACKSIZE);

   if ((dbproc = dbopen(login,server)) == NULL)
   {
      userlog("dbopen failed");
      return TRUE;
   };
   // Use the the right database
   dbuse(dbproc,database);
   dbcmd(dbproc,"select @@spid");
   dbsqlexec(dbproc);
   while (dbresults(dbproc) != NO_MORE_RESULTS)
   {
      dbbind(dbproc,1,SMALLBIND,(DBINT) 0,(BYTE *) spid);
      while (dbnextrow(dbproc) != NO_MORE_ROWS)
         ;
   };

   dbcmd(dbproc,"set nocount on");
   dbsqlexec(dbproc);
   while (dbresults(dbproc) != NO_MORE_RESULTS)
   {
      while (dbnextrow(dbproc) != NO_MORE_ROWS)
         ;
   };

   //rollback transaction on abort
   dbcmd(dbproc,"set XACT_ABORT ON");
   dbsqlexec(dbproc);
   while (dbresults(dbproc) != NO_MORE_RESULTS)
   {
      while (dbnextrow(dbproc) != NO_MORE_ROWS)
```

```
        ;
   };

   return(FALSE);

};   // SQLInit

//================================================================================
// FUNCTION: err_handler
//
//    Handles DB-Library errors
//
// ARGUMENTS:
//    DBPROCESS   *dbproc     DBPROCESS id pointer
//    int         severity    severity of error
//    int         dberr       error id
//    int         oserr       operating system specific error code
//    char        *dberrstr   printable error description of dberr
//    char        *oserrstr   printable error description of oserr
//
// RETURNS:
//    int      INT_CANCEL
//
// COMMENTS:   None
//
//================================================================================
int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr,
char *dberrstr, char *oserrstr)
{
   if ((dbproc == NULL) || (DBDEAD(dbproc)))
   {
      userlog("ErrHandler: DBPROC is invalid");
      return INT_CANCEL;
   };
   if (bFailed)
      return INT_CANCEL;
   if (oserr != DBNOERR)
   {
      sprintf(szWork,"ErrHandler: OSErr(%ld) - %s",oserr,oserrstr);
      userlog(szWork);
      bFailed = TRUE;
   };

   return INT_CANCEL;

};   // err_handler

//================================================================================
// FUNCTION: msg_handler
//
//    Handles DB-Library SQL Server error messages
//
// ARGUMENTS:
//    DBPROCESS   *dbproc     DBPROCESS id pointer
//    DBINT       msgno       message number
//    int         msgstate    message state
//    int         severity    message severity
//    char        *msgtext    printable message description
//
// RETURNS:    int   INT_CONTINUE   continue operation
//                   INT_CANCEL      cancel operation
//
// COMMENTS:   This function also sets the dead lock dbproc
//             variable if necessary.
//
//================================================================================
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext)
{

   if ((msgno == 5701) || (msgno == 2528) ||
       (msgno == 5703) || (msgno == 6006))
      return INT_CONTINUE;

   // deadlock message
   if (msgno == 1205)
   {
      // set the deadlock indicator
      bDeadlock = TRUE;
      return INT_CONTINUE;
   };

   if (bFailed)
      return INT_CANCEL;

   if (msgno == 0)
      return INT_CONTINUE;
   else
   {
      sprintf(szWork,"MsgHandler: MsgNo(%ld) - %s",msgno,msgtext);
      userlog(szWork);
      bFailed = TRUE;
   };

   return INT_CANCEL;

};   // msg_handler

//================================================================================
// FUNCTION: SQLStockLevel
//
//    Handles the stock level transaction.
//
// ARGUMENTS:
//    STOCK_LEVEL_DATA  StockLevel input / output data structure
//    dbdata (global)
//    bDeadlock (global)
//
// RETURNS:
//    SVC_NOERROR  success
//    !SVC_NOERROR failure
//
// COMMENTS:   None
//
//================================================================================
int SQLStockLevel(STOCK_LEVEL_DATA * psld)
{
   int tryit;
   short num_deadlocks = 0;
   RETCODE rc;
   BYTE * pData;
```

```
   bFailed = FALSE;
   bDeadlock = FALSE;

   for (tryit=0; tryit < DeadlockRetry; tryit++)
   {
      if (dbrpcinit(dbproc,"tpcc_stocklevel",0) == SUCCEED)
      {
         dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1,
            (BYTE *) &psld->w_id);
         dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1,
            (BYTE *) &psld->d_id);
         dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1,
            (BYTE *) &psld->thresh_hold);

         if (dbrpcexec(dbproc) == SUCCEED)
         {
            while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) &&
                   (rc != FAIL))
            {
               if (DBROWS(dbproc))
               {
                  while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) &&
                         (rc != FAIL))
                  {
                     if(pData=dbdata(dbproc,1))
                        psld->low_stock = *((long *) pData);
                  };
               };   // if (DBROWS(dbproc)
            };   // while (dbresults)
         };   // if (dbrpcexec)
      };   // if (dbrpcinit)
      if (bDeadlock)
      {
         num_deadlocks++;
         bDeadlock = FALSE;
         userlog("StockLevel Deadlock Retry (%d)",num_deadlocks);
         Sleep(10 * tryit);
      }
      else
      {
         strcpy(psld->execution_status,"Transaction commited.");
         return(SVC_NOERROR);
      };
   };  // for (tryit)

   // If we reached here, it means we quit after MAX_RETRY deadlocks
   strcpy(psld->execution_status,"Hit deadlock max.");
   userlog("StockLevel Deadlock Failure (%d)",num_deadlocks);
        return(SVCERR_DEADLOCK);

};   // SQLStockLevel

//============================================================================
// FUNCTION: SQLNewOrder
//
//    Handles the new order transaction.
//
// ARGUMENTS:
//    NEW_ORDER_DATA    NewOrder structure for input/output data
//    dbdata (global)
//    bDeadlock (global)
//
// RETURNS:
//    SVC_NOERROR  success
//    !SVC_NOERROR failure
//
// COMMENTS:   None
//
//============================================================================
int SQLNewOrder(NEW_ORDER_DATA * pnod)
{
   RETCODE rc;
   int i;
   DBINT commit_flag;
   short num_deadlocks = 0;
   int tryit;
   DBDATETIME datetime;
   BYTE * pData;

   bFailed = FALSE;
   bDeadlock = FALSE;

   for (tryit=0; tryit < DeadlockRetry; tryit++)
   {
      if (dbrpcinit(dbproc,"tpcc_neworder",0) == SUCCEED)
      {
         dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1,
            (BYTE *) &pnod->w_id);
         dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1,
            (BYTE *) &pnod->d_id);
         dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1,
            (BYTE *) &pnod->c_id);
         dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1,
            (BYTE *) &pnod->o_ol_cnt);

         pnod->o_all_local = 1;
         for (i = 0; i < pnod->o_ol_cnt; i++)
         {
            if (pnod->o_all_local &&
                pnod->Ol[i].ol_supply_w_id != pnod->w_id )
               pnod->o_all_local = 0;
         };
         dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1,
            (BYTE *) &pnod->o_all_local);

         for (i = 0; i < pnod->o_ol_cnt; i++)
         {
            dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1,
               (BYTE *) &pnod->Ol[i].ol_i_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1,
               (BYTE *) &pnod->Ol[i].ol_supply_w_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1,
               (BYTE *) &pnod->Ol[i].ol_quantity);
         };

         if (dbrpcexec(dbproc) == SUCCEED)
         {
            pnod->total_amount=0;
            // Get results from order line
            for (i = 0; i<pnod->o_ol_cnt; i++)
```

```
            {
               if (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) &&
                    (rc != FAIL))
               {
                  if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
                  {
                     while (dbnextrow(dbproc) != NO_MORE_ROWS)
                     {
                        if(pData=dbdata(dbproc, 1))
                           UtilStrCpy(pnod-
>Ol[i].ol_i_name,pData,dbdatlen(dbproc, 1));
                        if(pData=dbdata(dbproc, 2))
                           pnod->Ol[i].ol_stock = (*(DBSMALLINT *) pData);
                        if(pData=dbdata(dbproc, 3))
                           UtilStrCpy(pnod-
>Ol[i].ol_brand_generic,pData,dbdatlen(dbproc, 3));
                        if(pData=dbdata(dbproc, 4))
                           pnod->Ol[i].ol_i_price = (*(DBFLT8 *) pData);
                        if(pData=dbdata(dbproc, 5))
                           pnod->Ol[i].ol_amount = (*(DBFLT8 *) pData);
                        pnod->total_amount = pnod->total_amount + pnod-
>Ol[i].ol_amount;
                     };   // while (dbnextrow)
                  };   // if (DBROWS && dbnumcols)
               };   // if (dbresults)
            };   // for (o_ol_cnt)
            while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) &&
                   (rc != FAIL))
            {
               if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
               {
                  while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) &&
                         (rc != FAIL))
                  {
                     if(pData=dbdata(dbproc, 1))
                        pnod->w_tax = (*(DBFLT8 *) pData);
                     if(pData=dbdata(dbproc, 2))
                        pnod->d_tax = (*(DBFLT8 *) pData);
                     if(pData=dbdata(dbproc, 3))
                        pnod->o_id = (*(DBINT *) pData);
                     if(pData=dbdata(dbproc, 4))
                        UtilStrCpy(pnod->c_last,pData,dbdatlen(dbproc,4));
                     if(pData=dbdata(dbproc, 5))
                        pnod->c_discount = (*(DBFLT8 *) pData);
                     if(pData=dbdata(dbproc, 6))
                        UtilStrCpy(pnod-
>c_credit,pData,dbdatlen(dbproc,6));
                     if(pData=dbdata(dbproc, 7))
                     {
                        datetime = *((DBDATETIME *) pData);
                        dbdatecrack(dbproc,&pnod->o_entry_d,&datetime);
                     };
                     if(pData=dbdata(dbproc, 8))
                        commit_flag = (*(DBTINYINT *) pData);
                  };   // while (dbnextrow)
               };   // if (DBROWS && dbnumcols)
            };   // while (dbresults)
         };   // if (dbrpcexec)
      };   // if (dbrpcinit)
      if (bDeadlock)
      {
         num_deadlocks++;
         bDeadlock = FALSE;
         userlog("NewOrder Deadlock Retry (%d)",num_deadlocks);
         Sleep(10 * tryit);
      }
      else
      {
         if (commit_flag == 1)
         {
            pnod->total_amount = pnod->total_amount *
               ((1 + pnod->w_tax + pnod->d_tax) * (1 - pnod->c_discount));
            strcpy(pnod->execution_status,"Transaction commited.");
            return(SVC_NOERROR);
         }
         else
         {
            strcpy(pnod->execution_status,"Item number is not valid.");
            return(SVC_BADITEMID);
         };
      };   // !bDeadlock
   };   // for (tryit)

   // If we reached here, it means we quit after MAX_RETRY deadlocks
   strcpy(pnod->execution_status,"Hit deadlock max.");
   userlog("NewOrder Deadlock Failure (%d)",num_deadlocks);
   return(SVCERR_DEADLOCK);

};   // SQLNewOrder

//======================================================================
// FUNCTION: SQLPayment
//
//    Handles the payment transaction.
//
// ARGUMENTS:
//    PAYMENT_DATA      Payment input/output data structure
//    dbdata (global)
//    bDeadlock (global)
//
// RETURNS:
//    SVC_NOERROR  success
//    !SVC_NOERROR failure
//
// COMMENTS:   None
//
//======================================================================
int SQLPayment(PAYMENT_DATA *ppd)
{
   RETCODE rc;
   int tryit;
   short num_deadlocks = 0;
   DBDATETIME datetime;
   BYTE * pData;

   bFailed = FALSE;
   bDeadlock = FALSE;

   for (tryit=0; tryit < DeadlockRetry; tryit++)
   {
      if (dbrpcinit(dbproc,"tpcc_payment",0) == SUCCEED)
      {
```

```
         dbrpcparam(dbproc,NULL,0,SQLINT2,-1,-1,(BYTE *) &ppd->w_id);
         dbrpcparam(dbproc,NULL,0,SQLINT2,-1,-1,(BYTE *) &ppd->c_w_id);
         dbrpcparam(dbproc,NULL,0,SQLFLT8,-1,-1,(BYTE *) &ppd->h_amount);
         dbrpcparam(dbproc,NULL,0,SQLINT1,-1,-1,(BYTE *) &ppd->d_id);
         dbrpcparam(dbproc,NULL,0,SQLINT1,-1,-1,(BYTE *) &ppd->c_d_id);
         dbrpcparam(dbproc,NULL,0,SQLINT4,-1,-1,(BYTE *) &ppd->c_id);
         if (ppd->c_id == 0)
         {
            dbrpcparam(dbproc,NULL,0,SQLCHAR,-1,strlen(ppd->c_last),ppd-
>c_last);
         };
      };
      if (dbrpcexec(dbproc) == SUCCEED)
      {
         while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc !=
FAIL))
         {
            if (DBROWS(dbproc) && (dbnumcols(dbproc) == 27))
            {
               while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc !=
FAIL))
               {
                  if(pData=dbdata(dbproc,1))
                     ppd->c_id = *((DBINT *) pData);
                  if(pData=dbdata(dbproc,2))
                     UtilStrCpy(ppd->c_last,pData,dbdatlen(dbproc,2));
                  if(pData=dbdata(dbproc,3))
                  {
                     datetime = *((DBDATETIME *) pData);
                     dbdatecrack(dbproc,&ppd->h_date,&datetime);
                  };
                  if(pData=dbdata(dbproc,4))
                     UtilStrCpy(ppd->w_street_1,pData,dbdatlen(dbproc,4));
                  if(pData=dbdata(dbproc,5))
                     UtilStrCpy(ppd->w_street_2,pData,dbdatlen(dbproc,5));
                  if(pData=dbdata(dbproc,6))
                     UtilStrCpy(ppd->w_city,pData,dbdatlen(dbproc,6));
                  if(pData=dbdata(dbproc,7))
                     UtilStrCpy(ppd->w_state,pData,dbdatlen(dbproc,7));
                  if(pData=dbdata(dbproc,8))
                     UtilStrCpy(ppd->w_zip,pData,dbdatlen(dbproc,8));
                  if(pData=dbdata(dbproc,9))
                     UtilStrCpy(ppd->d_street_1,pData,dbdatlen(dbproc,9));
                  if(pData=dbdata(dbproc,10))
                     UtilStrCpy(ppd-
>d_street_2,pData,dbdatlen(dbproc,10));
                  if(pData=dbdata(dbproc,11))
                     UtilStrCpy(ppd->d_city,pData,dbdatlen(dbproc,11));
                  if(pData=dbdata(dbproc,12))
                     UtilStrCpy(ppd->d_state,pData,dbdatlen(dbproc,12));
                  if(pData=dbdata(dbproc,13))
                     UtilStrCpy(ppd->d_zip,pData,dbdatlen(dbproc,13));
                  if(pData=dbdata(dbproc,14))
                     UtilStrCpy(ppd->c_first,pData,dbdatlen(dbproc,14));
                  if(pData=dbdata(dbproc,15))
                     UtilStrCpy(ppd->c_middle,pData,dbdatlen(dbproc,15));
                  if(pData=dbdata(dbproc,16))
                     UtilStrCpy(ppd-
>c_street_1,pData,dbdatlen(dbproc,16));
                  if(pData=dbdata(dbproc,17))
                     UtilStrCpy(ppd-
>c_street_2,pData,dbdatlen(dbproc,17));
                  if(pData=dbdata(dbproc,18))
                     UtilStrCpy(ppd->c_city,pData,dbdatlen(dbproc,18));
                  if(pData=dbdata(dbproc,19))
                     UtilStrCpy(ppd->c_state,pData,dbdatlen(dbproc,19));
                  if(pData=dbdata(dbproc,20))
                     UtilStrCpy(ppd->c_zip,pData,dbdatlen(dbproc,20));
                  if(pData=dbdata(dbproc,21))
                     UtilStrCpy(ppd->c_phone,pData,dbdatlen(dbproc,21));
                  if(pData=dbdata(dbproc,22))
                  {
                     datetime = *((DBDATETIME *) pData);
                     dbdatecrack(dbproc,&ppd->c_since, &datetime);
                  };
                  if(pData=dbdata(dbproc,23))
                     UtilStrCpy(ppd->c_credit,pData,dbdatlen(dbproc,23));
                  if(pData=dbdata(dbproc,24))
                     ppd->c_credit_lim = (*(DBFLT8 *) pData);
                  if(pData=dbdata(dbproc,25))
                     ppd->c_discount = (*(DBFLT8 *) pData);
                  if(pData=dbdata(dbproc,26))
                     ppd->c_balance = (*(DBFLT8 *) pData);
                  if(pData=dbdata(dbproc,27))
                     UtilStrCpy(ppd->c_data,pData,dbdatlen(dbproc,27));
               };   // while (dbnextrow)
            };   // if (DBROWS && dbnumcols)
         };   // while (dbresults)
      };   // if (dbrpcexe)
      if (bDeadlock)
      {
         num_deadlocks++;
         bDeadlock = FALSE;
         userlog("Payment Deadlock Retry (%d)",num_deadlocks);
         Sleep(10 * tryit);
      }
      else
      {
         if (ppd->c_id == 0)
         {
            strcpy(ppd->execution_status,"Invalid Customer id,name.");
                           return(SVCERR_NOCUSTOMER);
         }
         else
            strcpy(ppd->execution_status,"Transaction commited.");
         return(SVC_NOERROR);
      };   // !bDeadlock
   };   // for (tryit)

   // If we reached here, it means we quit after MAX_RETRY deadlocks
   strcpy(ppd->execution_status,"Hit deadlock max.");
   userlog("Payment Deadlock Failure (%d)",num_deadlocks);
   return(SVCERR_DEADLOCK);

};   // SQLPayment

//================================================================
// FUNCTION: SQLOrderStatus
//
//    Handles the Order Status transaction.
```

```
//
// ARGUMENTS:
//    ORDER_STATUS_DATA      Payment input/output data structure
//    dbdata (global)
//    bDeadlock (global)
//
// RETURNS:
//    SVC_NOERROR  success
//    !SVC_NOERROR failure
//
// COMMENTS:   None
//
//=============================================================================
int SQLOrderStatus(ORDER_STATUS_DATA * posd)
{
   RETCODE rc;
   int tryit;
   short num_deadlocks = 0;
   int i;
   DBDATETIME datetime;
   BYTE * pData;

   bFailed = FALSE;
   bDeadlock = FALSE;

   for (tryit=0; tryit < DeadlockRetry; tryit++)
   {
      if (dbrpcinit(dbproc,"tpcc_orderstatus", 0) == SUCCEED)
      {
         dbrpcparam(dbproc,NULL,0,SQLINT2,-1,-1,(BYTE *) &posd->w_id);
         dbrpcparam(dbproc,NULL,0,SQLINT1,-1,-1,(BYTE *) &posd->d_id);
         dbrpcparam(dbproc,NULL,0,SQLINT4,-1,-1,(BYTE *) &posd->c_id);
         if (posd->c_id == 0)
         {
            dbrpcparam(dbproc,NULL,0,SQLCHAR,-1,strlen(posd->c_last),posd-
>c_last);
         };
      };
      if (dbrpcexec(dbproc) == SUCCEED)
      {
         while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc !=
FAIL))
         {
            if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
            {
               i = 0;
               while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc !=
FAIL))
               {
                  if(pData=dbdata(dbproc,1))
                     posd->OlOrderStatusData[i].ol_supply_w_id =
(*(DBSMALLINT *) pData);
                  if(pData=dbdata(dbproc,2))
                     posd->OlOrderStatusData[i].ol_i_id = (*(DBINT *)
pData);
                  if(pData=dbdata(dbproc,3))
                     posd->OlOrderStatusData[i].ol_quantity =
(*(DBSMALLINT *) pData);
                  if(pData=dbdata(dbproc,4))
                     posd->OlOrderStatusData[i].ol_amount = (*(DBFLT8 *)
pData);
                  if(pData=dbdata(dbproc,5))
                  {
                     datetime = *((DBDATETIME *) pData);
                     dbdatecrack(dbproc,&posd-
>OlOrderStatusData[i].ol_delivery_d,&datetime);
                  };
                  i++;
               };   // while (dbnextrow)
               posd->o_ol_cnt = i;
            }   // if (DBROWS && dbnumcols == 5)
            else
            if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
            {
               while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc !=
FAIL))
               {
                  if(pData=dbdata(dbproc,1))
                     posd->c_id = (*(DBINT *) pData);
                  if(pData=dbdata(dbproc,2))
                     UtilStrCpy(posd->c_last,pData,dbdatlen(dbproc,2));
                  if(pData=dbdata(dbproc,3))
                     UtilStrCpy(posd->c_first,pData,dbdatlen(dbproc,3));
                  if(pData=dbdata(dbproc,4))
                     UtilStrCpy(posd->c_middle,pData,dbdatlen(dbproc,4));
                  if(pData=dbdata(dbproc,5))
                  {
                     datetime = *((DBDATETIME *) pData);
                     dbdatecrack(dbproc,&posd->o_entry_d,&datetime);
                  };
                  if(pData=dbdata(dbproc,6))
                     posd->o_carrier_id = (*(DBSMALLINT *) pData);
                  if(pData=dbdata(dbproc,7))
                     posd->c_balance = (*(DBFLT8 *) pData);
                  if(pData=dbdata(dbproc,8))
                     posd->o_id = (*(DBINT *) pData);
               };   // while (dbnextrow)
            };   // if (DBROWS && dbnumcols == 8)
            if (i==0)
               return(SVCERR_NOORDERS); //"No orders found for customer"
         };   // while (dbresults)
      };   // if (dbrpcexec)
      if (bDeadlock)
      {
         num_deadlocks++;
         bDeadlock = FALSE;
         userlog("OrderStatus Deadlock Retry (%d)",num_deadlocks);
         Sleep(10 * tryit);
      }
      else
      {
         if (posd->c_id == 0 && posd->c_last[0] == 0)
         {
            strcpy(posd->execution_status,"Invalid Customer id,name.");
            return(SVCERR_NOCUSTOMER);
         }
         else
            strcpy(posd->execution_status,"Transaction commited.");
         return(SVC_NOERROR);
      };  // !bDeadlock
   };   // for (tryit)
```

```
   // If we reached here, it means we quit after MAX_RETRY deadlocks
   strcpy(posd->execution_status,"Hit deadlock max.");
   userlog("OrderStatus Deadlock Failure (%d)",num_deadlocks);
   return(SVCERR_DEADLOCK);

};   // SQLOrderStatus

//============================================================================
// FUNCTION: UtilStrCpy
//
//    Copies n characters from string pSrc to pDst and places a null
//    null character at the end of the destination string.  Unlike
//    strncpy this function ensures that the result string is always
//    null terminated.
//
//============================================================================
void UtilStrCpy(char * pDest, char * pSrc, int n)
{
   strncpy(pDest, pSrc, n);
   pDest[n] = '\0';
   return;
};   // UtilStrCpy

//============================================================================
//
// Function name: GetArgs
//
//============================================================================
VOID GetArgs(INT argc, CHAR **argv)
{
   INT j;
   CHAR * ptr;
   BOOL bRslt = TRUE;

   for (j = 1; j < argc; ++j)
   {
      ptr = argv[j];
      switch (ptr[1])
      {
         case 's':
         case 'S':
            strcpy(szServer,ptr+2);
            break;

         case 'd':
         case 'D':
            strcpy(szDatabase,ptr+2);
            break;

      };   // switch(ptr[1])
   };   // for (j = 1; j < argc; ++j)
};   // GetArgs
```

tpccdelv.c

```
// tpccdelv.c
//
// Copyright Unisys, 1997
// Copyright Microsoft, 1996


#include <windows.h>
#include <malloc.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>

#include <atmi.h>
#include <userlog.h>

#include "tpccsvr.h"

int   iServerNo = 0;
char   szServer[32]   = "tpccdelv";
char   szUser[32]            = { 0 };
char   szPassword[32] = { 0 };
char   szDatabase[32] = "tpcc";
char  szService[16]  = "tpccdelv";
char  szWork[200];

PDBPROCESS     dbproc;
int    spid;                                // spid assigned from dblib
BOOL  bFailed;
BOOL  bDeadlock;
short   DeadlockRetry  = (short)10;

FILE *fpLog;
char  szLogTitle[32];
BOOL  bFlush = FALSE;               // flush after every write

int err_handler(DBPROCESS *dbproc,int severity,int dberr,int oserr,
                char *dberrstr, char *oserrstr);
int msg_handler(DBPROCESS *dbproc,DBINT msgno,int msgstate,
                int severity,char *msgtext);
void WriteLog(DELIVERY_DATA * pdd);
BOOL OpenLogFile(void);
void CalculateElapsed(int * pElapsed,LPSYSTEMTIME lpBegin,
                      LPSYSTEMTIME lpEnd);
void UtilStrCpy(char * pDest, char * pSrc, int n);
void GetArgs(INT argc, CHAR **argv);

//============================================================================
//
// Function name: tpsvrinit
//
//============================================================================
tpsvrinit(int argc, char *argv[])
{
   GetArgs(argc,argv);
   if (iServerNo == 0)
   {
      userlog("Error - Server Number (-n option) Not Set");
      return(-1);
   };
   sprintf(szWork,"%s%ld Started, DBServer=%s,DB=%s",
      szService,iServerNo,szServer,szDatabase);
   userlog(szWork);
```

```
   if (OpenLogFile())
      return(-1);
   if (SQLInit(szServer,szDatabase,szUser,szPassword))
      return(-1);
   userlog("Database open, initialization complete");
   return(0);
};   // tpsvrinit

//======================================================================
//
// Function name: tpsvrdone
//
//======================================================================
void tpsvrdone()
{
   userlog("Shutdown request for tpccdelv server");
   if ( fpLog )
      fclose(fpLog);
   dbclose(dbproc);
   dbexit();
};   // tpsvrdone

//======================================================================
//
// Function name: DELIVERY
//
//    Entry point called by tuxedo for DELIVERY service requests.
//
//======================================================================
void DELIVERY(TPSVCINFO * svcinfo)
{
   int iRslt;
   DELIVERY_DATA * pdd;

   pdd = (DELIVERY_DATA *) svcinfo->data;
   iRslt = SQLDelivery(pdd);
   WriteLog(pdd);

   // Check for DBLib termination error
   if (bFailed)
   {
      strcpy(pdd->execution_status,szWork);
      userlog(szWork);
      tpreturn(TPFAIL,SVCERR_DBLIB,svcinfo->data,svcinfo->len,0);
   }
   else
   if (iRslt == 0)
      tpreturn(TPSUCCESS,0,svcinfo->data,svcinfo->len,0);
   else
      tpreturn(TPFAIL,iRslt,svcinfo->data,svcinfo->len,0);

};   // DELIVERY

//======================================================================
//
// Function name: SQLInit
//
//    Set global dbproc and spid.
//
//    Result:
//       FALSE - database open, dbproc valid
//       TRUE - database open failed
//
//======================================================================
BOOL SQLInit(CHAR * pSvr,CHAR * pDB,CHAR * pUsr,CHAR * pPW,CHAR * pSvc)
{
   char szApp[32];
   char server[256];
   char database[256];
   char user[256];
   char password[256];
   LOGINREC    *login;

   dbinit();
   // install error and message handlers
   dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
   dberrhandle((DBERRHANDLE_PROC)err_handler);

   dbproc = NULL;
   strcpy(server,pSvr);
   strcpy(database,pDB);
   strcpy(user,pUsr);
   strcpy(password,pPW);
   sprintf(szApp,"%s%ld",pSvc,_getpid());

   login = dblogin();
   if (!*user )
      DBSETLUSER(login,"sa");
   else
      DBSETLUSER(login,user);
   DBSETLPWD(login,password);
   DBSETLHOST(login,szApp);
// DBSETLPACKET(login,(unsigned short)DEFCLPACKSIZE);

   if ((dbproc = dbopen(login,server)) == NULL)
   {
      userlog("dbopen failed");
      return TRUE;
   };
   // Use the the right database
   dbuse(dbproc,database);
   dbcmd(dbproc,"select @@spid");
   dbsqlexec(dbproc);
   while (dbresults(dbproc) != NO_MORE_RESULTS)
   {
      dbbind(dbproc,1,SMALLBIND,(DBINT) 0,(BYTE *) spid);
      while (dbnextrow(dbproc) != NO_MORE_ROWS)
         ;
   };

   dbcmd(dbproc,"set nocount on");
   dbsqlexec(dbproc);
   while (dbresults(dbproc) != NO_MORE_RESULTS)
   {
      while (dbnextrow(dbproc) != NO_MORE_ROWS)
         ;
   };

   //rollback transaction on abort
   dbcmd(dbproc,"set XACT_ABORT ON");
   dbsqlexec(dbproc);
   while (dbresults(dbproc) != NO_MORE_RESULTS)
```

```
   {
      while (dbnextrow(dbproc) != NO_MORE_ROWS)
         ;
   };

   return(FALSE);

};   // SQLInit

//=========================================================================
// FUNCTION: err_handler
//
//    Handles DB-Library errors
//
// ARGUMENTS:
//    DBPROCESS   *dbproc     DBPROCESS id pointer
//    int         severity    severity of error
//    int         dberr       error id
//    int         oserr       operating system specific error code
//    char        *dberrstr   printable error description of dberr
//    char        *oserrstr   printable error description of oserr
//
// RETURNS:
//    int      INT_CANCEL
//
// COMMENTS:   None
//
//=========================================================================
int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr,
char *dberrstr, char *oserrstr)
{
   if ((dbproc == NULL) || (DBDEAD(dbproc)))
   {
      userlog("ErrHandler: DBPROC is invalid");
      return INT_CANCEL;
   };
   if (bFailed)
      return INT_CANCEL;
   if (oserr != DBNOERR)
   {
      sprintf(szWork,"ErrHandler: OSErr(%ld) - %s",oserr,oserrstr);
      userlog(szWork);
      bFailed = TRUE;
   };

   return INT_CANCEL;

};   // err_handler

//=========================================================================
// FUNCTION: msg_handler
//
//    Handles DB-Library SQL Server error messages
//
// ARGUMENTS:
//    DBPROCESS   *dbproc     DBPROCESS id pointer
//    DBINT       msgno       message number
//    int         msgstate    message state
//    int         severity    message severity
//    char        *msgtext    printable message description
//
// RETURNS:    int   INT_CONTINUE   continue operation
//                   INT_CANCEL      cancel operation
//
// COMMENTS:   This function also sets the dead lock dbproc
//             variable if necessary.
//
//=========================================================================
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext)
{

   if ((msgno == 5701) || (msgno == 2528) ||
       (msgno == 5703) || (msgno == 6006))
      return INT_CONTINUE;

   // deadlock message
   if (msgno == 1205)
   {
      // set the deadlock indicator
      bDeadlock = TRUE;
      return INT_CONTINUE;
   };

   if (bFailed)
      return INT_CANCEL;

   if (msgno == 0)
      return INT_CONTINUE;
   else
   {
      sprintf(szWork,"MsgHandler: MsgNo(%ld) - %s",msgno,msgtext);
      userlog(szWork);
      bFailed = TRUE;
   };

   return INT_CANCEL;

};   // msg_handler

//=========================================================================
// FUNCTION: SQLDelivery
//
// ARGUMENTS:
//    pdd      delivery transaction structure
//    dbdata (global)
//    bDeadlock (global)
//
// RETURNS:
//    SVC_NOERROR  success
//    !SVC_NOERROR failure
//
// COMMENTS:   None
//
//=========================================================================
int SQLDelivery(DELIVERY_DATA * pdd)
{
   RETCODE rc;
   int i;
   short num_deadlocks = 0;
```

```
   int tryit;
   DBDATETIME datetime;
   BYTE * pData;

   bFailed = FALSE;
   bDeadlock = FALSE;
   pdd->iComplete = 0;

   for (tryit=0; tryit < DeadlockRetry; tryit++)
   {
      if (dbrpcinit(dbproc,"tpcc_delivery",0) == SUCCEED)
      {
         dbrpcparam(dbproc,NULL,0,SQLINT2,-1,-1,(BYTE *) &pdd->w_id);
         dbrpcparam(dbproc,NULL,0,SQLINT1,-1,-1,(BYTE *) &pdd-
>o_carrier_id);

         if (dbrpcexec(dbproc) == SUCCEED)
         {
            while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc !=
FAIL))
            {
               while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc !=
FAIL))
               {
                  for (i = 0; i < 10; i++)
                  {
                     if(pData = dbdata(dbproc,i + 1))
                        pdd->o_id[i] = *((DBINT *)pData);
                     else
                        pdd->o_id[i] = 0;
                  };
               };   // while (dbnextrow)
            };   // while (dbresults)
         };   // if (dbrpcexec)
      };   // if (dbrpcinit)
      if (bDeadlock)
      {
         num_deadlocks++;
         bDeadlock = FALSE;
         userlog("Delivery Deadlock Retry (%d)",num_deadlocks);
         Sleep(10 * tryit);
      }
      else
      {
         GetLocalTime(&pdd->EndTime);
         pdd->iComplete = 1;
         strcpy(pdd->execution_status,"Transaction commited.");
         return(SVC_NOERROR);
      };
   };  // for (tryit)

   // If we reached here, it means we quit after MAX_RETRY deadlocks
   strcpy(pdd->execution_status,"Hit deadlock max.");
   userlog("Delivery Deadlock Failure (%d)",num_deadlocks);
   return(SVCERR_DEADLOCK);

};   // SQLDelivery

//======================================================================
// FUNCTION: WriteLog
//
//    Writes the delivery results to a log file.
//
// ARGUMENTS:
//    pDelivery   delivery information.
//
// RETURNS:
//
// COMMENTS:
//    Record format:
//       QTime,EndTime,Elapsed,w_id,o_carrier_id,o_id1, ... o_id10
//
//======================================================================
void WriteLog(DELIVERY_DATA * pdd)
{
   int elapsed = 9999999;
   if (pdd->iComplete)
      CalculateElapsed(&elapsed,&pdd->QTime,&pdd->EndTime);
   fprintf(fpLog,
      "%2.2d/%2.2d/%2.2d,%2.2d:%2.2d:%2.2d:%3.3d,%2.2d:%2.2d:%2.2d:%3.3d,"
      "%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
      pdd->EndTime.wYear - 1900,pdd->EndTime.wMonth,pdd->EndTime.wDay,
      pdd->QTime.wHour,pdd->QTime.wMinute,
      pdd->QTime.wSecond,pdd->QTime.wMilliseconds,
      pdd->EndTime.wHour,pdd->EndTime.wMinute,
      pdd->EndTime.wSecond,pdd->EndTime.wMilliseconds,
      elapsed,pdd->w_id,pdd->o_carrier_id,
      pdd->o_id[0],pdd->o_id[1],pdd->o_id[2],pdd->o_id[3],pdd->o_id[4],
      pdd->o_id[5],pdd->o_id[6],pdd->o_id[7],pdd->o_id[8],pdd->o_id[9] );
       if (bFlush)
             fflush(fpLog);
};   // WriteLog

//======================================================================
// FUNCTION: OpenLogFile
//
//    Opens the delivery log file.
//
// ARGUMENTS:
//    None.
//
// RETURNS:
//    FALSE    Log file successfully opened
//    TRUE     Failed to open log file
//
// COMMENTS:
//
//======================================================================
BOOL OpenLogFile(void)
{
   sprintf(szLogTitle,"%s%ld",LOGFILE_NAME,iServerNo);
   fpLog = fopen(szLogTitle,"ab");
      if (!fpLog)
   {
      sprintf(szWork,"LogFile %s Open Failed (%ld)",
         szLogTitle,GetLastError());
      userlog(szWork);
      return(TRUE);
   };
   return(FALSE);
};   // OpenLogFile
```

```
//====================================================================
// FUNCTION: CalculateElapsed
//
//    Calculates the elapsed time of the delivery transaction.
//
// ARGUMENTS:
//    lpBegin  time delivery was queued
//    lpEnd            time delivery update completed
//
// RETURNS:
//    int            pElapsed elapsed time result (in milliseconds)
//
// COMMENTS:
//    None
//
//====================================================================
void CalculateElapsed(int * pElapsed,LPSYSTEMTIME lpBegin,
                      LPSYSTEMTIME lpEnd)
{
   int tmBegin;
   int tmEnd;

   tmBegin = (lpBegin->wHour * 3600000) + (lpBegin->wMinute * 60000) +
      (lpBegin->wSecond * 1000) + lpBegin->wMilliseconds;
   tmEnd = (lpEnd->wHour * 3600000) + (lpEnd->wMinute * 60000) +
      (lpEnd->wSecond * 1000) + lpEnd->wMilliseconds;
   *pElapsed = tmEnd - tmBegin;

   // Check for day boundry, this will function for 24 hour period but
   // will fail over a 48 hours period.
   if (*pElapsed < 0)
      *pElapsed = *pElapsed + (24 * 60 * 60 * 1000);
   return;
};   // CalculateElapsed

//====================================================================
// FUNCTION: UtilStrCpy
//
//    Copies n characters from string pSrc to pDst and places a null
//    null character at the end of the destination string.
//
// ARGUMENTS:
//    char     *pDest   destination string pointer
//    char     *pSrc    source string pointer
//    int      n        number of characters to copy
//
// RETURNS:           None
//
// COMMENTS:
//    Unlike strncpy this function ensures that the result string is
//    always null terminated.
//
//====================================================================
void UtilStrCpy(char * pDest, char * pSrc, int n)
{
   strncpy(pDest, pSrc, n);
   pDest[n] = '\0';
   return;
};   // UtilStrCpy
```

```
//====================================================================
//
// Function name: GetArgs
//
//====================================================================
void GetArgs(INT argc, CHAR **argv)
{
   INT j;
   CHAR * ptr;
   BOOL bRslt = TRUE;

   for (j = 1; j < argc; ++j)
   {
      ptr = argv[j];
      switch (ptr[1])
      {
         case 's':
         case 'S':
            strcpy(szServer,ptr+2);
            break;

         case 'd':
         case 'D':
            strcpy(szDatabase,ptr+2);
            break;

         case 'n':
         case 'N':
            iServerNo = atoi(ptr+2);
            break;

         case 'F':
         case 'f':
            bFlush = TRUE;    //turn on delilog flush when written.
            break;

      };   // switch(ptr[1])
   };   // for (j = 1; j < argc; ++j)
};   // GetArgs
```

DELIVERY REPORT MAKEFILE

```
# Microsoft Developer Studio Generated NMAKE File, Format Version 4.20
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Console Application" 0x0103

!IF "$(CFG)" == ""
CFG=delirpt - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to delirpt - Win32 Debug.
!ENDIF

!IF "$(CFG)" != "delirpt - Win32 Release" && "$(CFG)" !=\
 "delirpt - Win32 Debug"
!MESSAGE Invalid configuration "$(CFG)" specified.
!MESSAGE You can specify a configuration when running NMAKE on this
makefile
!MESSAGE by defining the macro CFG on the command line.  For example:
!MESSAGE
```

```
!MESSAGE NMAKE /f "delirpt.mak" CFG="delirpt - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "delirpt - Win32 Release" (based on "Win32 (x86) Console
Application")
!MESSAGE "delirpt - Win32 Debug" (based on "Win32 (x86) Console
Application")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
################################################################################
######
# Begin Project
CPP=cl.exe
RSC=rc.exe

!IF  "$(CFG)" == "delirpt - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "delirpt_"
# PROP BASE Intermediate_Dir "delirpt_"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "delirpt_"
# PROP Intermediate_Dir "delirpt_"
# PROP Target_Dir ""
OUTDIR=.\delirpt_
INTDIR=.\delirpt_

ALL : "$(OUTDIR)\delirpt.exe"

CLEAN :
       -@erase "$(INTDIR)\DELIRPT.OBJ"
       -@erase "$(OUTDIR)\delirpt.exe"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"
/YX /c
# ADD CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /YX /c
CPP_PROJ=/nologo /ML /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"\
 /Fp"$(INTDIR)/delirpt.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\delirpt_/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/delirpt.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib /nologo /subsystem:console /machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib /nologo /subsystem:console /incremental:no\
 /pdb:"$(OUTDIR)/delirpt.pdb" /machine:I386 /out:"$(OUTDIR)/delirpt.exe"
LINK32_OBJS= \
       "$(INTDIR)\DELIRPT.OBJ"

"$(OUTDIR)\delirpt.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF  "$(CFG)" == "delirpt - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : "$(OUTDIR)\delirpt.exe"

CLEAN :
       -@erase "$(INTDIR)\DELIRPT.OBJ"
       -@erase "$(INTDIR)\vc40.idb"
       -@erase "$(INTDIR)\vc40.pdb"
       -@erase "$(OUTDIR)\delirpt.exe"
       -@erase "$(OUTDIR)\delirpt.ilk"
       -@erase "$(OUTDIR)\delirpt.pdb"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE" /YX /c
# ADD CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_CONSOLE"
/YX /c
CPP_PROJ=/nologo /MLd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE"\
 /Fp"$(INTDIR)/delirpt.pch" /YX /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
CPP_OBJS=.\Debug/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
```

```
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/delirpt.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /debug /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib /nologo /subsystem:console /debug /machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib /nologo /subsystem:console /incremental:yes\
 /pdb:"$(OUTDIR)/delirpt.pdb" /debug /machine:I386
/out:"$(OUTDIR)/delirpt.exe"
LINK32_OBJS= \
        "$(INTDIR)\DELIRPT.OBJ"

"$(OUTDIR)\delirpt.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

################################################################################
######
# Begin Target

# Name "delirpt - Win32 Release"
# Name "delirpt - Win32 Debug"

!IF  "$(CFG)" == "delirpt - Win32 Release"

!ELSEIF  "$(CFG)" == "delirpt - Win32 Debug"

!ENDIF

################################################################################
######
# Begin Source File

SOURCE=.\DELIRPT.C

"$(INTDIR)\DELIRPT.OBJ" : $(SOURCE) "$(INTDIR)"


# End Source File
# End Target
# End Project
################################################################################
######
```

delirpt.c

```
/*      FILE:           DELIRPT.C
 *                              Microsoft TPC-C Kit Ver. 3.00.000
 *
 *                              Copyright Microsoft, 1996
 *
 *      PURPOSE:        Delivery report processing application
 *      Author:         Philip Durr
 *                              philipdu@Microsoft.com
 */

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

#define LOGFILE_READ_EOF      0
                        //check log file flag return current state
#define LOGFILE_CLEAR_EOF     1
                        //clear end of log file flag
#define LOGFILE_SET_EOF               2
                                //set flag end of log file reached

#define INTERVAL                      .01
                                //90th percentile calculation bucket
interval

#define ERR_SUCCESS                                             1000
                        //success no error
#define ERR_READING_LOGFILE                             1001
                //io errors occured reading delivery log file
#define ERR_INSUFFICIENT_MEMORY                         1002
                //insuficient memory to process 90th percentile report
#define ERR_CANNOT_OPEN_RESULTS_FILE 1005
        //Cannot open delivery results file delilog.

typedef struct _RPTLINE
{
        SYSTEMTIME      start;
                                //delilog report line start time
        SYSTEMTIME      end;
                                //delilog report line end time
        int                     response;
                                        //delilog report line time delivery
took in milliseconds
```

```
	int			w_id;
				//delilog report line warehouse id
for delivery
	int			o_carrier_id;
				//delilog report line carier id for delivery
	int			items[10];
				//delilog report line delivery line
items
} RPTLINE, *PRPTLINE;

//error message structure used in ErrorMessage API
typedef struct _SERRORMSG
{
	int		iError;			//error id of message
	char	szMsg[80];		//message to sent to browser
} SERRORMSG;

int			versionMS = 4;
		//delirpt version
int			versionMM = 0;
int			versionLS = 0;
int			iReport;
			//delirpt report to process
int			iStartTime;
			//begin times to accept for report
int			iEndTime;
			//end times to accept for report
FILE		*fpLog;
		//log file stream
CHAR szLogFileTitle[100];
#define DEFAULTLOGTITLE "delilog."

//Local function prototypes
void		main(int argc, char *argv[]);
static int	Init(void);
static void	Restore(void);
static int	DoReport(void);
int			AverageResponse(void);
int			SkippedDelivery(void);
int			Percentile90th(void);
BOOL		CheckTimes(PRPTLINE pRptLine);
static int	OpenLogFile(void);
static void	CloseLogFile(void);
static void	ResetLogFile(void);
static BOOL	LogEOF(int iOperation);
static BOOL	ReadReportLine(char *szBuffer, PRPTLINE pRptLine);
static BOOL	ParseReportLine(char *szLine, PRPTLINE pRptLine);
static BOOL	ParseDate(char *szDate, LPSYSTEMTIME pTime);
static BOOL	ParseTime(char *szTime, LPSYSTEMTIME pTime);
static void	ErrorMessage(int iError);
static BOOL	GetParameters(int argc, char *argv[]);
static void	PrintParameters(void);
static void	PrintHeader(void);
static void	cls(void);
static BOOL	IsNumeric(char *ptr);


/* FUNCTION: int main(int argc, char *argv[])
 *
 * PURPOSE:	This function is the beginning execution point for the
delivery executable.
 *
 * ARGUMENTS:  int		argc	number of command line arguments
passed to delivery
 *				char	*argv[] array of command line
argument pointers
 *
 * RETURNS:		None
 *
 * COMMENTS:	None
 *
 */

void main(int argc, char *argv[])
{
	int	iError;

	PrintHeader();

	if ( GetParameters(argc, argv) )
	{
		PrintParameters();
		return;
	}

	if ( (iError=Init()) != ERR_SUCCESS )
	{
		ErrorMessage(iError);
		Restore();
		return;
	}

	if ( (iError = DoReport()) != ERR_SUCCESS )
		ErrorMessage(iError);

	Restore();

	return;
}

/* FUNCTION: static int Init(void)
 *
 * PURPOSE:	This function initializes the delirtp application.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:		None
 *
 * COMMENTS:	None
 *
 */

static int Init(void)
{
	int iError;

	if ( (iError = OpenLogFile()) )
		return iError;
	return TRUE;
}

/* FUNCTION: static void Restore(void)
```

```
 *
 * PURPOSE:    This function cleans up the delirpt application before
termination.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:           None
 *
 * COMMENTS:   None
 *
 */

static void Restore(void)
{
       CloseLogFile();
       return;
}

/* FUNCTION: static int DoReport(void)
 *
 * PURPOSE:           This function dispatches the requested report.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:           ERR_SUCCESS if successfull or error code if an
error occurs.
 *
 * COMMENTS:   None
 *
 */

static int DoReport(void)
{
       int iRc;

       switch(iReport)
       {
              case 1:
                     iRc = AverageResponse();
                     break;
              case 2:
                     iRc = Percentile90th();
                     break;
              case 3:
                     iRc = SkippedDelivery();
                     break;
              case 4:
                     if ( (iRc = AverageResponse()) != ERR_SUCCESS )
                            break;
                     if ( (iRc = Percentile90th()) != ERR_SUCCESS )
                            break;
                     if ( (iRc = SkippedDelivery()) != ERR_SUCCESS )
                            break;
                     break;
       }
       return iRc;
}

/* FUNCTION: int AverageResponse(void)
 *
 * PURPOSE:           This function processes the AverageResponse report.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:           ERR_SUCCESS if successfull or error code if an
error occurs.
 *
 * COMMENTS:   None
 *
 */

int AverageResponse(void)
{
       RPTLINE reportLine;
       int            iTotalResponse;
       int            iLines;
       double  fAverage;
       char    szDelivery[128];

       ResetLogFile();

       iTotalResponse = 0;
       iLines = 0;
       printf("\n\n******** Average Response Time Report *******\n");
       while ( !LogEOF(LOGFILE_READ_EOF) )
       {
              if ( ReadReportLine(szDelivery, &reportLine) )
                     return ERR_READING_LOGFILE;
              if ( !LogEOF(LOGFILE_READ_EOF) )
              {
                     if ( CheckTimes(&reportLine) )
                            continue;
                     iLines++;
                     iTotalResponse += reportLine.response;

                     if ( iLines % 10 == 0 )
                            printf("Reading Report Line:\t%d\r",
iLines);
              }
       }
       printf("                                              \r");
       if ( iLines == 0 )
       {
              printf("No deliveries found.\n");
       }
       else
       {
              fAverage = ((double)iTotalResponse /
(double)iLines)/(double)1000;
              printf("Total Deliveries:      %10.0f\n", (float)iLines);
              printf("Total Response Times:  %10.3f\n",
((float)iTotalResponse/(float)1000));
              printf("Average Response Time: %10.3f\n", fAverage);
       }

       return ERR_SUCCESS;
}

/* FUNCTION: int Percentile90th(void)
 *
```

```
 * PURPOSE:           This function processes the 90th percentile report.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:           ERR_SUCCESS if successfull or error code if an
error occurs.
 *
 * COMMENTS:   This function requires enough space to allocate needed
 *                           buckets which will be 2 * max response time
in
 *                           deci-seconds.
 *
 */

int Percentile90th(void)
{
       RPTLINE reportLine;
       int             iBucketSize;
       int             i;
       int             iResponseSeconds;
       int             iMaxSeconds;
       int             iTotalBuckets;
       double  iTotal;
       double  i90thPercent;
       short   *psBuckets;
       char    szDelivery[128];

       printf("\n\n******** 90th Percentile *******\n");
       printf("Calculating Max Response Seconds...\n");

       ResetLogFile();

       iMaxSeconds = -1;
       while ( !LogEOF(LOGFILE_READ_EOF) )
       {
              if ( ReadReportLine(szDelivery, &reportLine) )
                     return ERR_READING_LOGFILE;
              if ( szDelivery[0] == '*' )
                     continue;
              if ( !LogEOF(LOGFILE_READ_EOF) )
              {
                     if ( iMaxSeconds < reportLine.response )
                            iMaxSeconds = reportLine.response;
              }
       }

   iTotalBuckets = iMaxSeconds + 1;

       printf("Allocating Buckets...\n");

       iBucketSize = iTotalBuckets * sizeof(short);

       if ( !(psBuckets = (short *)malloc(iBucketSize)) )
              return ERR_INSUFFICIENT_MEMORY;

       ZeroMemory(psBuckets, iBucketSize);

       iTotal = 0;

       ResetLogFile();
       printf("Calculating Distribution...\n");

       iMaxSeconds = -1;
       while ( !LogEOF(LOGFILE_READ_EOF) )
       {
              if ( ReadReportLine(szDelivery, &reportLine) )
                     return ERR_READING_LOGFILE;
              if ( szDelivery[0] == '*' )
                     continue;
              if ( !LogEOF(LOGFILE_READ_EOF) )
              {
                     if ( CheckTimes(&reportLine) )
                            continue;
                     psBuckets[reportLine.response]++;
                     iTotal++;
                     if ( iMaxSeconds < reportLine.response )
                            iMaxSeconds = reportLine.response;
              }
       }

       printf("Max Response Time = %d.%d\n",
      (iMaxSeconds / 1000),(iMaxSeconds % 1000));

       i90thPercent = iTotal * .9;

       for(i=0, iTotal = 0.0; iTotal < i90thPercent; iTotal +=
(double)psBuckets[i] )
              i++;

       printf("90th Percentile = %d.%d\n", i/1000, (i % 1000));

       free(psBuckets);

       return ERR_SUCCESS;
}

/* FUNCTION: int SkippedDelivery(void)
 *
 * PURPOSE:           This function processes the Skipped Deliveries
report.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:           ERR_SUCCESS if successfull or error code if an
error occurs.
 *
 * COMMENTS:   None
 *
 */

int SkippedDelivery(void)
{
       RPTLINE reportLine;
       char    szDelivery[128];
       int             i;
       int             items[10];

       ResetLogFile();

       printf("\n\n******** Skipped Delivery Report *******\n");
       memset(items, 0, sizeof(items));
       printf("Reading Delivery Log File...");
```

```
        while ( !LogEOF(LOGFILE_READ_EOF) )
        {
                if ( ReadReportLine(szDelivery, &reportLine) )
                        return ERR_READING_LOGFILE;
                if ( !LogEOF(LOGFILE_READ_EOF) )
                {
                        if ( CheckTimes(&reportLine) )
                                continue;
                        for(i=0; i<10; i++)
                        {
                                if ( !reportLine.items[i] )
                                        items[i]++;
                        }
                }
        }
        printf("\n");
        printf("Skipped delivery table.\n");
        printf("  1    2    3    4    5    6    7    8    9   10 \n");
        printf("---- ---- ---- ---- ---- ---- ---- ---- ---- ----\n");
        for(i=0; i<10; i++)
                printf("%4.4d ", items[i]);
        printf("\n");

        return ERR_SUCCESS;
}

/* FUNCTION: BOOL CheckTimes(PRPTLINE pRptLine)
 *
 * PURPOSE:    This function checks to see of the delilog record falls
withing the
 *                      begin and end time from the command line.
 *
 * ARGUMENTS:  PRPTLINE       pRptLine       delilog processed report
line.
 *
 * RETURNS:            BOOL    FALSE   if report line is not within the
 *                                                      requested
start and end times.
 *                                      TRUE    if the report line is
within the
 *                                                      requested
start and end times.
 *
 * COMMENTS:   If startTime and endTime are both 0 then the user requested
 *                      the default behavior which is all records in
delilog are
 *                      valid.
 */

BOOL CheckTimes(PRPTLINE pRptLine)
{
        int     iRptEndTime;
        int     iRptStartTime;

        iRptStartTime = (pRptLine->start.wHour * 3600000) + (pRptLine-
>start.wMinute * 60000) + (pRptLine->start.wSecond * 1000) + pRptLine-
>start.wMilliseconds;

        iRptEndTime = (pRptLine->end.wHour * 3600000) + (pRptLine-
>end.wMinute * 60000) + (pRptLine->end.wSecond * 1000) + pRptLine-
>end.wMilliseconds;

        if ( iStartTime == 0 && iEndTime == 0 )
                return FALSE;

        if ( iStartTime <= iRptStartTime && iEndTime >= iRptEndTime )
                return FALSE;

        return TRUE;
}

/* FUNCTION: int OpenLogFile(void)
 *
 * PURPOSE:    This function opens the delivery log file for use.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:            int     ERR_CANNOT_OPEN_RESULTS_FILE  Cannot create
results log file.
 *                                      ERR_SUCCESS
 *                      Log file successfully opened
 *
 *
 * COMMENTS:   None
 *
 */

static int OpenLogFile(void)
{

        fpLog = fopen(szLogFileTitle, "rb");

        if ( !fpLog )
                return ERR_CANNOT_OPEN_RESULTS_FILE;

        return ERR_SUCCESS;
}

/* FUNCTION: int CloseLogFile(void)
 *
 * PURPOSE:    This function closes the delivery log file.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:            None
 *
 * COMMENTS:   None
 *
 */

static void CloseLogFile(void)
{
        if ( fpLog )
                fclose(fpLog);

        return;
}
```

```
/* FUNCTION: static void ResetLogFile(void)
 *
 * PURPOSE:    This function prepares the delilog. file for reading
 *
 * ARGUMENTS:  None
 *
 * RETURNS:            None
 *
 * COMMENTS:   None
 *
 */

static void ResetLogFile(void)
{
       fseek(fpLog, 0L, SEEK_SET);
       LogEOF(LOGFILE_CLEAR_EOF);

       return;
}

/* FUNCTION: static BOOL LogEOF(int iOperation)
 *
 * PURPOSE:    This function tracks and reports the end of file condition
 *                     on the delilog file.
 *
 * ARGUMENTS:  int iOperation requested operation this can be:
 *
       LOGFILE_READ_EOF      check log file flag return current state
 *
       LOGFILE_CLEAR_EOF     clear end of log file flag
 *
       LOGFILE_SET_EOF               set flag end of log file reached
 *
 *
 * RETURNS:            None
 *
 * COMMENTS:   None
 *
 */

static BOOL LogEOF(int iOperation)
{
       static BOOL bEOF;

       switch(iOperation)
       {
              case LOGFILE_READ_EOF:
                     return bEOF;
                     break;
              case LOGFILE_CLEAR_EOF:
                     bEOF = FALSE;
                     break;
              case LOGFILE_SET_EOF:
                     bEOF = TRUE;
                     break;
       }
       return FALSE;
}

/* FUNCTION: static BOOL ReadReportLine(char *szBuffer, PRPTLINE pRptLine)
 *
 * PURPOSE:    This function reads a text line from the delilog file.
 *                     on the delilog file.
 *
 * ARGUMENTS:  char           *szBuffer      buffer to placed read delilog
file line into.
 *                            PRPTLINE       pRptLine       returned
structure containing parsed delilog
 *
       report line.
 *
 * RETURNS:            FALSE   if successfull or TRUE if an error occurs.
 *
 * COMMENTS:   None
 *
 */

static BOOL ReadReportLine(char *szBuffer, PRPTLINE pRptLine)
{
       int i = 0;
       int ch;
       int iEof;

       while( i < 128 )
       {
              ch = fgetc(fpLog);
              if ( iEof = feof(fpLog) )
                     break;
              if ( ch == '\r' )
              {
                     if ( i )
                            break;
                     continue;
              }
              if ( ch == '\n' )
                     continue;
              szBuffer[i++] = ch;

       }

       //delivery item format is to long cannot be a valid delivery item
       if ( i >= 128 )
              return TRUE;

       szBuffer[i] = 0;
       if ( iEof )
       {
              LogEOF(LOGFILE_SET_EOF);
              if ( i == 0 )
                     return FALSE;
       }
       return ParseReportLine(szBuffer, pRptLine);
}

/* FUNCTION: static BOOL ParseReportLine(char *szLine, PRPTLINE pRptLine)
 *
 * PURPOSE:    This function reads a text line from the delilog file.
 *                     on the delilog file.
 *
 * ARGUMENTS:  char           *szLine        buffer containing the delilog
file line to be parsed.
```

```
 *                            PRPTLINE       pRptLine       returned
structure containing parsed delilog
 *
       report line values.
 *
 * RETURNS:            FALSE   if successfull or TRUE if an error occurs.
 *
 * COMMENTS:   None
 *
 */

static BOOL ParseReportLine(char *szLine, PRPTLINE pRptLine)
{
       int i;

       if ( ParseDate(szLine, &pRptLine->start) )
              return TRUE;

       pRptLine->end.wYear = pRptLine->start.wYear;
       pRptLine->end.wMonth = pRptLine->start.wMonth;
       pRptLine->end.wDay = pRptLine->start.wDay;

       if ( !(szLine = strchr(szLine, ',')) )
              return TRUE;
       szLine++;

       if ( ParseTime(szLine, &pRptLine->start) )
              return TRUE;

       if ( !(szLine = strchr(szLine, ',')) )
              return TRUE;
       szLine++;

       if ( ParseTime(szLine, &pRptLine->end) )
              return TRUE;

       if ( !(szLine = strchr(szLine, ',')) )
              return TRUE;
       szLine++;

       if ( !IsNumeric(szLine) )
              return TRUE;
       pRptLine->response = atoi(szLine);

       if ( !(szLine = strchr(szLine, ',')) )
              return TRUE;
       szLine++;

       if ( !IsNumeric(szLine) )
              return TRUE;
       pRptLine->w_id = atoi(szLine);

       if ( !(szLine = strchr(szLine, ',')) )
              return TRUE;
       szLine++;

       if ( !IsNumeric(szLine) )
              return TRUE;
       pRptLine->o_carrier_id = atoi(szLine);

       if ( !(szLine = strchr(szLine, ',')) )
              return TRUE;
       szLine++;

       for(i=0; i<10; i++)
       {
              if ( !IsNumeric(szLine) )
                     return TRUE;
              pRptLine->items[i] = atoi(szLine);

              if ( i<9 && !(szLine = strchr(szLine, ',')) )
                     return TRUE;
              szLine++;
       }

       return FALSE;
}

/* FUNCTION: static BOOL ParseDate(char *szDate, LPSYSTEMTIME pTime)
 *
 * PURPOSE:    This function validates and extracts a date string in the
format
 *                     yy/mm/dd into an SYSTEMTIME structure.
 *
 * ARGUMENTS:  char                   *szDate        buffer containing the
date to be parsed.
 *                            LPSYSTEMTIME   pTime          system time
structure where date will be placed.
 *
 * RETURNS:            FALSE if successfull or TRUE if an error occurs.
 *
 * COMMENTS:   None
 *
 */

static BOOL ParseDate(char *szDate, LPSYSTEMTIME pTime)
{
       if ( !isdigit(*szDate) || !isdigit(*(szDate+1)) || *(szDate+2) !=
'/' ||
                !isdigit(*(szDate+3)) || !isdigit(*(szDate+4)) ||
*(szDate+5) != '/' ||
                !isdigit(*(szDate+6)) || !isdigit(*(szDate+7)) )
                return TRUE;

       pTime->wYear = atoi(szDate);

       pTime->wMonth = atoi(szDate+3);

       pTime->wDay = atoi(szDate+6);

       if ( pTime->wMonth > 12 || pTime->wMonth < 0 || pTime->wDay > 31
|| pTime->wDay < 0 )
              return TRUE;

       return FALSE;
}

/* FUNCTION: static BOOL ParseTime(char *szTime, LPSYSTEMTIME pTime)
 *
```

```
 * PURPOSE:    This function validates and extracts a time string in the
format
 *                     hh:mm:ss:mmm into an SYSTEMTIME structure.
 *
 * ARGUMENTS:  char                  *szTime        buffer containing the
time to be parsed.
 *                           LPSYSTEMTIME   pTime          system time
structure where date will be placed.
 *
 * RETURNS:            FALSE if successfull or TRUE if an error occurs.
 *
 * COMMENTS:   None
 *
 */

static BOOL ParseTime(char *szTime, LPSYSTEMTIME pTime)
{
       if ( !isdigit(*szTime) || !isdigit(*(szTime+1)) || *(szTime+2) !=
':' ||
              !isdigit(*(szTime+3)) || !isdigit(*(szTime+4)) ||
*(szTime+5) != ':' ||
               !isdigit(*(szTime+6)) || !isdigit(*(szTime+7)) ||
*(szTime+8) != ':' ||
               !isdigit(*(szTime+9)) || !isdigit(*(szTime+10)) ||
!isdigit(*(szTime+11)) )
                return TRUE;

       pTime->wHour = atoi(szTime);
       pTime->wMinute = atoi(szTime+3);
       pTime->wSecond = atoi(szTime+6);
       pTime->wMilliseconds  = atoi(szTime+9);

       if ( pTime->wHour > 23 || pTime->wHour < 0 ||
              pTime->wMinute > 59 || pTime->wMinute < 0 ||
              pTime->wSecond > 59 || pTime->wSecond < 0 ||
              pTime->wMilliseconds < 0 )
              return TRUE;

       if ( pTime->wMilliseconds > 999 )
       {
              pTime->wSecond += (pTime->wMilliseconds/1000);
              pTime->wMilliseconds = pTime->wMilliseconds  % 1000;
       }

       return FALSE;
}

/* FUNCTION: void ErrorMessage(int iError)
 *
 * PURPOSE:    This function displays an error message in the delivery
executable's console window.
 *
 * ARGUMENTS:  int           iError  error id to be displayed
 *
 * RETURNS:           None
 *
 * COMMENTS:   None
 *
 */

static void ErrorMessage(int iError)
{
       int i;

       static SERRORMSG errorMsgs[] =
       {
              {      ERR_SUCCESS,
              "Success, no error."
                                                  },
              {      ERR_CANNOT_OPEN_RESULTS_FILE,
       "Cannot open delivery results log file."
              },
              {      ERR_READING_LOGFILE,
       "Reading delivery log file, Delivery item format incorrect."
       },
              {      ERR_INSUFFICIENT_MEMORY,
       "insufficient memory to process 90th percentile report."
       },
              {      0,
                            ""

              }
       };

       for(i=0; errorMsgs[i].szMsg[0]; i++)
       {
              if ( iError == errorMsgs[i].iError )
              {
                     printf("\nError(%d): %s\n", iError,
errorMsgs[i].szMsg);
                     return;
              }
       }
       printf("Error(%d): %s", errorMsgs[0].szMsg);
       return;
}

/* FUNCTION: BOOL GetParameters(int argc, char *argv[])
 *
 * PURPOSE:    This function parses the command line passed in to the
delivery executable, initializing
 *                     and filling in global variable parameters.
 *
 * ARGUMENTS:  int            argc    number of command line arguments
passed to delivery
 *                            char    *argv[] array of command line
argument pointers
 *
 * RETURNS:           BOOL    FALSE   parameter read successfull
 *                                                  TRUE    user has requested
parameter information screen be displayed.
 *
 * COMMENTS:   None
 *
 */

static BOOL GetParameters(int argc, char *argv[])
{
       int                    i;
       SYSTEMTIME     startTime;
       SYSTEMTIME     endTime;
   UINT uLogTitleLen;
```

```
        iStartTime = 0;
        iEndTime = 0;
        iReport = 4;
    strcpy(szLogFileTitle,DEFAULTLOGTITLE);

        for(i=0; i<argc; i++)
        {
                if ( argv[i][0] == '-' || argv[i][0] == '/' )
                {
                        switch(argv[i][1])
                        {
                                case 'S':
                                case 's':
                                        if ( ParseTime(argv[i]+2,
&startTime) )
                                                return TRUE;
                                        iStartTime = (startTime.wHour *
3600000) + (startTime.wMinute * 60000) + (startTime.wSecond * 1000) +
startTime.wMilliseconds;
                                        break;
                                case 'E':
                                case 'e':
                                        if ( ParseTime(argv[i]+2, &endTime)
)
                                                return TRUE;
                                        iEndTime = (endTime.wHour * 3600000)
+ (endTime.wMinute * 60000) + (endTime.wSecond * 1000) +
endTime.wMilliseconds;
                                        break;
                                case 'R':
                                case 'r':
                                        iReport = atoi(argv[i]+2);
                                        if ( iReport > 4 || iReport < 1 )
                                                iReport = 4;
                                        break;
                                case 'F':
                                case 'f':
                uLogTitleLen = strlen(argv[i] - 2);
                if (uLogTitleLen > 0 && uLogTitleLen <
sizeof(szLogFileTitle))
                {
                   strcpy(szLogFileTitle,argv[i]+2);
                   printf("Log File Title set to %s",szLogFileTitle);
                };
                                        break;
                                case '?':
                                        return TRUE;
                        }
                }
        }
        return FALSE;
}

/* FUNCTION: void PrintParameters(void)
 *
 * PURPOSE:    This function displays the supported command line flags.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:            None
 *
 * COMMENTS:   None
 *
 */

static void PrintParameters(void)
{
        PrintHeader();
        printf("DELIRPT:\n\n");
        printf("Parameter
Default\n");
        printf("-----------------------------------------------------------
-------------\n");
        printf("-S Start Time HH:MM:SS:MMM
All    \n");
        printf("-E End Time HH:MM:SS:MMM
All    \n");
        printf("-R 1)Average Response, 2)90th 3) Skipped 4) All
All    \n");
        printf("-? This help screen\n\n");
        printf("Note:  Command line switches are NOT case sensitive.\n");

        return;
}

/* FUNCTION: void PrintHeader(void)
 *
 * PURPOSE:    This function displays the delivery report applications
banner information.
 *
 * ARGUMENTS:  None
 *
 * RETURNS:            None
 *
 * COMMENTS:   None
 *
 */

static void PrintHeader(void)
{
        //cls();

        printf("************************************************\n");
        printf("*                                              *\n");
        printf("*  Microsoft SQL Server 6.5                    *\n");
        printf("*                                              *\n");
        printf("*  HTML TPC-C BENCHMARK KIT: Delivery Report   *\n");
        printf("*  Version %d.%2.2d.%3.3d
*\n", versionMS, versionMM, versionLS);
        printf("*                                              *\n");
        printf("************************************************\n\n");

        return;
}

/* FUNCTION: void cls(void)
 *
 * PURPOSE:    This function clears the console window
 *
```

```
 * ARGUMENTS:  None
 *
 * RETURNS:           None
 *
 * COMMENTS:   None
 *
 */

static void cls(void)
{
       HANDLE  hConsole;
       COORD   coordScreen = { 0, 0 };                    //here's where
we'll home the cursor
       DWORD   cCharsWritten;
       CONSOLE_SCREEN_BUFFER_INFO    csbi;          //to get buffer info
       DWORD                                        dwConSize;
       //number of character cells in the current buffer

       hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

       //get the number of character cells in the current buffer

       GetConsoleScreenBufferInfo( hConsole, &csbi );
       dwConSize = csbi.dwSize.X * csbi.dwSize.Y;

       //fill the entire screen with blanks
       FillConsoleOutputCharacter( hConsole, (TCHAR) ' ', dwConSize,
coordScreen, &cCharsWritten );
       GetConsoleScreenBufferInfo( hConsole, &csbi );

       //now set the buffer's attributes accordingly
       FillConsoleOutputAttribute( hConsole, csbi.wAttributes,dwConSize,
coordScreen, &cCharsWritten );

       //put the cursor at (0, 0)
       SetConsoleCursorPosition( hConsole, coordScreen );

       return;
}

/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE:    This function determines if a string is numeric. It fails
if any characters other
 *                    than numeric and null terminator are present.
 *
 * ARGUMENTS:  char                  *ptr    pointer to string to check.
 *
 * RETURNS:           BOOL    FALSE   if string is not all numeric
 *                                    TRUE    if string contains
only numeric characters i.e. '0' - '9'
 *
 * COMMENTS:   A comma is counted as a valid delimiter.
 *
 */

static BOOL IsNumeric(char *ptr)
{
       if ( *ptr == 0 )
              return FALSE;
       while( *ptr && isdigit(*ptr) )
              ptr++;
       if ( !*ptr || *ptr == ',' )
              return TRUE;
       else
              return FALSE;
}
```

Appendix B - Database Design

Build Scripts

CREATEDB.SQL

```
/*  TPC-C Benchmark Kit                             */
/*                                                  */
/*  CREATEDB.SQL                                    */
/*                                                  */
/*  This script is used to create the database      */
/*  for a 1110 warehouse tpcc database.                 */


use master
go

if exists ( select name from sysdatabases where name = "tpcc" )
        drop database tpcc
go

/* tpcc database size                             */
/*         3,500 MB on misc segment               */
/*        63,000 MB on cs segment                 */
/*        22,500 MB on ol segment                 */
/*        32,750 MB on syslogs segment                */
/* total 121,750 MB (118.896 GB)                  */

/* total of 17 device fragments                      */

create database tpcc

        on tpc_misc1 = 655,   tpc_misc2 = 655,
           tpc_misc3 = 655,   tpc_misc4 = 655,
           tpc_misc5 = 880,

           tpc_cs1 =  9952,   tpc_cs2 =  9952,
           tpc_cs3 =  9952,   tpc_cs4 =  9952,
           tpc_cs5 =  9952,   tpc_cs6 = 13240,

           tpc_ol1 =  4549,   tpc_ol2 =  4549,
           tpc_ol3 =  4549,   tpc_ol4 =  4549,
           tpc_ol5 =  4304

    log on tpc_log = 32750

go
```

DBOPT1.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  DBOPT1.SQL
*/
/*
*/
/*  Set database options for database load
*/


use master
go

sp_dboption tpcc,'select into/bulkcopy',true
go

sp_dboption tpcc,'trunc. log on chkpt.',true
go

use tpcc
go

checkpoint
go

use tpcc_admin
go

sp_dboption tpcc,'trunc. log on chkpt.',true
go
```

DBOPT2.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  DBOPT2.SQL
*/
/*
*/
/*  Reset database options after database load
*/
```

```
use master
go

sp_dboption tpcc,'select ',false
go

sp_dboption tpcc,'trunc. ',false
go

use tpcc
go

checkpoint
go
```

DISKINIT.SQL

```
/*  TPC-C Benchmark Kit                              */
/*                                                   */
/*  DISKINIT.SQL                                     */
/*                                                   */
/*  This script is used create devices               */
/*  for 1110 warehouse tpcc database          */


use master
go


/* devices for warehouse, district, item, history,       */
/*            orders and new-order tables               */
/*   655MB = 335,360 pages of 2KB per device                 */
/*   880MB = 450,560 pages of 2KB per device                 */
/* 3,500MB total (3.418GB) = 4 x 655 + 880          */

disk init name = "tpc_misc1",
       physname   = "G:",
       vdevno     = 14,
       size       = 335360
go

disk init name = "tpc_misc2",
       physname   = "H:",
       vdevno     = 15,
       size       = 335360
go

disk init name = "tpc_misc3",
       physname   = "I:",
       vdevno     = 16,
       size       = 335360
go

disk init name = "tpc_misc4",
       physname   = "F:",
       vdevno     = 17,
       size       = 335360
go

disk init name = "tpc_misc5",
       physname   = "E:",
       vdevno     = 18,
       size       = 450560
go


/* devices for customer and stock tables                    */
/*  9,952MB = 5,095,424 pages of 2KB per device             */
/* 13,240MB = 6,778,880 pages of 2KB per device             */
/* 63,000MB total (61.523GB) = 5 x 9952 + 13240                */

disk init name = "tpc_cs1",
       physname = "L:",
       vdevno   = 20,
       size     = 5095424
go

disk init name = "tpc_cs2",
       physname = "M:",
       vdevno   = 21,
       size     = 5095424
go

disk init name = "tpc_cs3",
       physname = "N:",
       vdevno   = 22,
       size     = 5095424
go

disk init name = "tpc_cs4",
       physname = "O:",
       vdevno   = 23,
       size     = 5095424
go

disk init name = "tpc_cs5",
       physname = "J:",
       vdevno   = 24,
       size     = 5095424
go

disk init name = "tpc_cs6",
       physname = "K:",
       vdevno   = 25,
       size     = 6778880
go


/* devices for order-line                                 */
/*  4,549MB =  2,329,088 pages of 2KB per volume          */
/*  4,304MB =  2,203,648 pages of 2KB per volume          */
/* 22,500MB total (21.973GB) = 4 x 4549 + 4304                */

disk init name = "tpc_ol1",
       physname = "T:",
       vdevno   = 27,
       size     = 2329088
go

disk init name = "tpc_ol2",
       physname = "U:",
```

```
        vdevno   = 28,
        size     = 2329088
go

disk init name = "tpc_ol3",
        physname = "R:",
        vdevno   = 29,
        size     = 2329088
go

disk init name = "tpc_ol4",
        physname = "S:",
        vdevno   = 30,
        size     = 2329088
go

disk init name = "tpc_ol5",
        physname = "P:",
        vdevno   = 31,
        size     = 2203648
go


/* Log device                                              */
/* 32,750MB = 16,768,000 pages of 2KB                   */
/* 32,750MB total (31.982GB)                            */

disk init name = "tpc_log",
        physname   = "V:",
        vdevno     = 33,
        size       = 16768000
go
```

IDXCUSCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXCUSCL.SQL
*/
/*
*/
/*  Creates clustered index on customer (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where name = 'customer_c1' )
        drop index customer.customer_c1
go

select getdate()
go
create unique clustered index customer_c1 on customer(c_w_id, c_d_id,
c_id)
        with sorted_data on cs_seg
go
select getdate()
go
```

IDXCUSNC.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXCUSNC.SQL
*/
/*
*/
/*  Creates non-clustered index on customer (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where  name = 'customer_nc1' )
        drop index customer.customer_nc1
go

select getdate()
go
create unique nonclustered index customer_nc1 on customer(c_w_id, c_d_id,
c_last, c_first, c_id)
        on cs_seg
go
select getdate()
go
```

IDXDISCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXDISCL.SQL
*/
/*
*/
/*  Creates clustered index on district (seg)
*/
```

```
use tpcc
go


if exists ( select name from sysindexes where name = 'district_c1' )
       drop index district.district_c1
go

select getdate()
go
create unique clustered index  district_c1 on district(d_w_id, d_id)
       with fillfactor=1 on misc_seg
go
select getdate()
go
```

IDXITMCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXITMCL.SQL
*/
/*
*/
/*  Creates clustered index on item (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where  name = 'item_c1' )
       drop index item.item_c1
go

select getdate()
go
create unique clustered index item_c1 on item(i_id)
       with sorted_data on misc_seg
go
select getdate()
go
```

IDXNODCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXNODCL.SQL
*/
/*
*/
/*  Creates clustered index on new-order (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where  name = 'new_order_c1' )
       drop index new_order.new_order_c1
go

select getdate()
go
create unique clustered index new_order_c1 on new_order(no_w_id, no_d_id,
no_o_id)
       with sorted_data on misc_seg
go
select getdate()
go
```

IDXODLCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXODLCL.SQL
*/
/*
*/
/*  Creates clustered index on order-line (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where  name = 'order_line_c1' )
       drop index order_line.order_line_c1
go

select getdate()
go
create unique clustered index order_line_c1 on order_line(ol_w_id,
ol_d_id, ol_o_id, ol_number)
       with sorted_data on ol_seg
go
select getdate()
go
```

IDXORDCL.SQL

```
/*  TPC-C Benchmark Kit
*/
```

```
/*
*/
/*  IDXORDCL.SQL
*/
/*
*/
/*  Creates clustered index on orders (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where  name = 'orders_c1' )
        drop index orders.orders_c1
go

select getdate()
go
create unique clustered index orders_c1 on orders(o_w_id, o_d_id, o_id)
        with sorted_data on misc_seg
go
select getdate()
go
```

IDXSTKCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXSTKCL.SQL
*/
/*
*/
/*  Creates clustered index on stock (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where  name = 'stock_c1' )
        drop index stock.stock_c1
go

select getdate()
go
create unique clustered index stock_c1 on stock(s_i_id, s_w_id)
        with sorted_data on cs_seg
go
select getdate()
go
```

IDXWARCL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXWARCL.SQL
*/
/*
*/
/*  Creates clustered index on warehouse (seg)
*/


use tpcc
go


if exists ( select name from sysindexes where name = 'warehouse_c1' )
        drop index warehouse.warehouse_c1
go

select getdate()
go
create unique clustered index warehouse_c1 on warehouse(w_id)
        with fillfactor=1 on misc_seg
go
select getdate()
go
```

PINTABLE.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  PINTABLE.SQL
*/
/*
*/
/*  This script file is used to 'pin' certain tables in the data cache
*/

use tpcc
go

exec sp_tableoption "district","pintable",true
exec sp_tableoption "warehouse","pintable",true
exec sp_tableoption "new_order","pintable",true
exec sp_tableoption "item","pintable",true
go
```

SEGMENT.SQL

```
/*  TPC-C Benchmark Kit                          */
/*                                               */
```

```
/*  SEGMENT.SQL                                     */
/*                                                  */
/*  This script is used create segments             */

use tpcc
go

exec sp_dropsegment misc_seg
go
exec sp_dropsegment cs_seg
go
exec sp_dropsegment ol_seg
go


/* create segment for miscellaneous tables (warehouse, */
/* district, item, orders, new_order, and history)     */

sp_addsegment    misc_seg, tpc_misc1
go
sp_extendsegment misc_seg, tpc_misc2
go
sp_extendsegment misc_seg, tpc_misc3
go
sp_extendsegment misc_seg, tpc_misc4
go
sp_extendsegment misc_seg, tpc_misc5
go

/* create segment for customer and stock tables */

sp_addsegment    cs_seg, tpc_cs1
go
sp_extendsegment cs_seg, tpc_cs2
go
sp_extendsegment cs_seg, tpc_cs3
go
sp_extendsegment cs_seg, tpc_cs4
go
sp_extendsegment cs_seg, tpc_cs5
go
sp_extendsegment cs_seg, tpc_cs6
go

/* create segment for order-line table */

sp_addsegment    ol_seg, tpc_ol1
go
sp_extendsegment ol_seg, tpc_ol2
go
sp_extendsegment ol_seg, tpc_ol3
go
sp_extendsegment ol_seg, tpc_ol4
go
sp_extendsegment ol_seg, tpc_ol5
go
```

TABLES .SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  TABLES.SQL
*/
/*
*/
/*  Creates TPC-C tables (seg)
*/


use tpcc
go

checkpoint
go


if exists ( select name from sysobjects where name = 'warehouse' )
       drop table warehouse
go

create table warehouse
(
       w_id                          smallint,
       w_name                        char(10),
       w_street_1                    char(20),
       w_street_2                    char(20),
       w_city                        char(20),
       w_state                       char(2),
       w_zip                         char(9),
       w_tax                         numeric(4,4),
       w_ytd                         numeric(12,2)
) on misc_seg
go


if exists ( select name from sysobjects where name = 'district' )
       drop table district
go

create table district
(
       d_id                          tinyint,
       d_w_id                        smallint,
       d_name                        char(10),
       d_street_1                    char(20),
       d_street_2                    char(20),
       d_city                        char(20),
       d_state                       char(2),
       d_zip                         char(9),
       d_tax                         numeric(4,4),
       d_ytd                         numeric(12,2),
       d_next_o_id                   int
) on misc_seg
```

```
go


if exists ( select name from sysobjects where name = 'customer' )
       drop table customer
go

create table customer
(
       c_id                          int,
       c_d_id                        tinyint,
       c_w_id                        smallint,
       c_first                       char(16),
       c_middle                      char(2),
       c_last                        char(16),
       c_street_1                    char(20),
       c_street_2                    char(20),
       c_city                        char(20),
       c_state                       char(2),
       c_zip                         char(9),
       c_phone                       char(16),
       c_since                       datetime,
       c_credit                      char(2),
       c_credit_lim          numeric(12,2),
       c_discount                    numeric(4,4),
       c_balance                     numeric(12,2),
       c_ytd_payment         numeric(12,2),
       c_payment_cnt         smallint,
       c_delivery_cnt        smallint,
       c_data_1                      char(250),
       c_data_2                      char(250)
) on cs_seg
go


if exists ( select name from sysobjects where name = 'history' )
       drop table history
go

create table history
(
       h_c_id                        int,
       h_c_d_id                      tinyint,
       h_c_w_id                      smallint,
       h_d_id                        tinyint,
       h_w_id                        smallint,
       h_date                        datetime,
       h_amount                      numeric(6,2),
       h_data                        char(24)
) on misc_seg
go


if exists ( select name from sysobjects where name = 'new_order' )
       drop table new_order
go

create table new_order
(
       no_o_id                       int,
       no_d_id                       tinyint,
       no_w_id                       smallint
) on misc_seg
go


if exists ( select name from sysobjects where name = 'orders' )
       drop table orders
go

create table orders
(
       o_id                          int,
       o_d_id                        tinyint,
       o_w_id                        smallint,
       o_c_id                        int,
       o_entry_d                     datetime,
       o_carrier_id          tinyint,
       o_ol_cnt                      tinyint,
       o_all_local                   tinyint
) on misc_seg
go


if exists ( select name from sysobjects where name = 'order_line' )
       drop table order_line
go

create table order_line
(
       ol_o_id                       int,
       ol_d_id                       tinyint,
       ol_w_id                       smallint,
       ol_number                     tinyint,
       ol_i_id                       int,
       ol_supply_w_id        smallint,
       ol_delivery_d         datetime,
       ol_quantity                   smallint,
       ol_amount                     numeric(6,2),
       ol_dist_info          char(24)
) on ol_seg
go


if exists ( select name from sysobjects where name = 'item' )
       drop table item
go

create table item
(
       i_id                          int,
       i_im_id                       int,
       i_name                        char(24),
       i_price                       numeric(5,2),
       i_data                        char(50)
) on misc_seg
go


if exists ( select name from sysobjects where name = 'stock' )
```

```
	drop table stock
go

create table stock
(
	s_i_id			int,
	s_w_id			smallint,
	s_quantity		smallint,
	s_dist_01		char(24),
	s_dist_02		char(24),
	s_dist_03		char(24),
	s_dist_04		char(24),
	s_dist_05		char(24),
	s_dist_06		char(24),
	s_dist_07		char(24),
	s_dist_08		char(24),
	s_dist_09		char(24),
	s_dist_10		char(24),
	s_ytd			int,
	s_order_cnt		smallint,
	s_remote_cnt	smallint,
	s_data			char(50)
) on cs_seg
go
```

TPCCBCP.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  TPCCBCP.SQL
*/
/*
*/
/*  This script file sets the table lock option for bulk load
*/


use tpcc
go

exec sp_tableoption "warehouse","table lock on bulk load",true
exec sp_tableoption "district","table lock on bulk load",true
exec sp_tableoption "stock","table lock on bulk load",true
exec sp_tableoption "item","table lock on bulk load",true
exec sp_tableoption "customer","table lock on bulk load",true
exec sp_tableoption "history","table lock on bulk load",true
exec sp_tableoption "orders","table lock on bulk load",true
exec sp_tableoption "order_line","table lock on bulk load",true
exec sp_tableoption "new_order","table lock on bulk load",true
go
```

TPCCIRL.SQL

```
/*  TPC-C Benchmark Kit
*/
/*
*/
/*  TPCCIRL.SQL
*/
/*
*/
/*  This script file sets the insert row lock option on selected tables
*/


use tpcc
go

exec sp_tableoption "history","insert row lock",true
exec sp_tableoption "new_order","insert row lock",true
exec sp_tableoption "orders","insert row lock",true
exec sp_tableoption "order_line","insert row lock",true
go
```

WARMUP.SQL

```
/* Warm-up TPC-C database */

use tpcc
go

update sysobjects set cache= 2 from sysobjects where name='stock'
go

update sysobjects set cache= 5 from sysobjects where name='customer'
go

select name, id, cache from sysobjects where id > 100
go

dbcc gaminit
go

select "tpcc database GAMINIT finished!"
go
```

Stored Procedures

NEWORD.SQL

```
/*  File:       NEWORD.SQL
*/
/*              Microsoft TPC-C Kit Ver. 3.00.000
*/
/*              Audited 08/23/96, By Francois Raab
*/
/*
*/
/*              Copyright Microsoft, 1996
*/
/*
*/
```

```
/*  Purpose:    New-Order transaction for Microsoft TPC-C Benchmark Kit
*/
/*  Author:     Damien Lindauer
*/
/*              damienl@Microsoft.com
*/


use tpcc
go

/* new-order transaction stored procedure */

if exists ( select name from sysobjects where name = "tpcc_neworder" )
       drop procedure tpcc_neworder
go

/*  Modified by rick vicik, 2/4/97  */
/*  Combined initialization of local variables into district update
statement  */
/*  Combined 3 huge case select statements into a single one  */

create proc tpcc_neworder
                                                    @w_id
smallint,
                                                    @d_id
tinyint,
                                                    @c_id           int,
                                                    @o_ol_cnt
tinyint,
                                                    @o_all_local
tinyint,
                                                    @i_id1  int = 0,
@s_w_id1 smallint = 0, @ol_qty1 smallint = 0,
                                                    @i_id2  int = 0,
@s_w_id2 smallint = 0, @ol_qty2 smallint = 0,
                                                    @i_id3  int = 0,
@s_w_id3 smallint = 0, @ol_qty3 smallint = 0,
                                                    @i_id4  int = 0,
@s_w_id4 smallint = 0, @ol_qty4 smallint = 0,
                                                    @i_id5  int = 0,
@s_w_id5 smallint = 0, @ol_qty5 smallint = 0,
                                                    @i_id6  int = 0,
@s_w_id6 smallint = 0, @ol_qty6 smallint = 0,
                                                    @i_id7  int = 0,
@s_w_id7 smallint = 0, @ol_qty7 smallint = 0,
                                                    @i_id8  int = 0,
@s_w_id8 smallint = 0, @ol_qty8 smallint = 0,
                                                    @i_id9  int = 0,
@s_w_id9 smallint = 0, @ol_qty9 smallint = 0,
                                                    @i_id10 int = 0,
@s_w_id10 smallint = 0, @ol_qty10 smallint = 0,
                                                    @i_id11 int = 0,
@s_w_id11 smallint = 0, @ol_qty11 smallint = 0,
                                                    @i_id12 int = 0,
@s_w_id12 smallint = 0, @ol_qty12 smallint = 0,
                                                    @i_id13 int = 0,
@s_w_id13 smallint = 0, @ol_qty13 smallint = 0,
                                                    @i_id14 int = 0,
@s_w_id14 smallint = 0, @ol_qty14 smallint = 0,
                                                    @i_id15 int = 0,
@s_w_id15 smallint = 0, @ol_qty15 smallint = 0


as
declare @w_tax          numeric(4,4),
        @d_tax          numeric(4,4),
        @c_last         char(16),
        @c_credit       char(2),
        @c_discount     numeric(4,4),
        @i_price        numeric(5,2),
        @i_name         char(24),
        @i_data         char(50),
        @o_entry_d      datetime,
        @remote_flag    int,
        @s_quantity     smallint,
        @s_data         char(50),
        @s_dist         char(24),
               @li_no           int,
               @o_id                    int,
               @commit_flag     int,
        @li_id          int,
        @li_s_w_id      smallint,
        @li_qty         smallint,
               @ol_number               int,
               @c_id_local              int

begin

begin transaction n

/*  get district tax and next availible order id and update */
/*  plus initialize local variables  */

    update district
       set @d_tax      = d_tax,
           @o_id       = d_next_o_id,
               d_next_o_id = d_next_o_id + 1,
        @o_entry_d = getdate(),
        @li_no=0,
        @commit_flag = 1
           where d_w_id = @w_id and
                 d_id   = @d_id

       /*  process orderlines */
    while (@li_no < @o_ol_cnt)
       begin

        select @li_no = @li_no + 1

           /*  Set i_id, s_w_id, and qty for this lineitem */

           select @li_id = case @li_no
                   when 1 then @i_id1
                          when 2 then @i_id2
                          when 3 then @i_id3
                          when 4 then @i_id4
                          when 5 then @i_id5
                          when 6 then @i_id6
                          when 7 then @i_id7
```

```
                    when 8 then @i_id8
                    when 9 then @i_id9
                    when 10 then @i_id10
                    when 11 then @i_id11
                    when 12 then @i_id12
                    when 13 then @i_id13
                    when 14 then @i_id14
                    when 15 then @i_id15
                    end,

                    @li_s_w_id = case @li_no
                    when 1 then @s_w_id1
                    when 2 then @s_w_id2
                    when 3 then @s_w_id3
                    when 4 then @s_w_id4
                    when 5 then @s_w_id5
                    when 6 then @s_w_id6
                    when 7 then @s_w_id7
                    when 8 then @s_w_id8
                    when 9 then @s_w_id9
                    when 10 then @s_w_id10
                    when 11 then @s_w_id11
                    when 12 then @s_w_id12
                    when 13 then @s_w_id13
                    when 14 then @s_w_id14
                    when 15 then @s_w_id15
                    end,

                    @li_qty = case  @li_no
                    when 1 then @ol_qty1
                    when 2 then @ol_qty2
                    when 3 then @ol_qty3
                    when 4 then @ol_qty4
                    when 5 then @ol_qty5
                    when 6 then @ol_qty6
                    when 7 then @ol_qty7
                    when 8 then @ol_qty8
                    when 9 then @ol_qty9
                    when 10 then @ol_qty10
                    when 11 then @ol_qty11
                    when 12 then @ol_qty12
                    when 13 then @ol_qty13
                    when 14 then @ol_qty14
                    when 15 then @ol_qty15
                    end

    /* get item data (no one updates item) */

    select @i_price = i_price,
           @i_name  = i_name,
           @i_data  = i_data
    from item (tablock holdlock)
       where i_id = @li_id

      /* if there actually is an item with this id, go to work */

           if (@@rowcount > 0)
           begin
        update stock set s_ytd        = s_ytd + @li_qty,
                         @s_quantity  = s_quantity,
                         s_quantity   = s_quantity - @li_qty +
                                        case when (s_quantity - @li_qty < 10)
    then 91 else 0 end,
                         s_order_cnt  = s_order_cnt + 1,
                         s_remote_cnt = s_remote_cnt + case
                             when (@li_s_w_id = @w_id) then 0 else 1
    end,
                         @s_data      = s_data,
                         @s_dist      = case @d_id
                                           when 1  then s_dist_01
                                        when 2  then s_dist_02
                                        when 3  then s_dist_03
                                        when 4  then s_dist_04
                                        when 5  then s_dist_05
                                        when 6  then s_dist_06
                                        when 7  then s_dist_07
                                        when 8  then s_dist_08
                                        when 9  then s_dist_09
                                        when 10 then s_dist_10
                                    end
                from stock
                where s_i_id = @li_id and
                      s_w_id = @li_s_w_id

                    /* insert order_line data (using data from item and
    stock) */

                insert into order_line values(@o_id,              /* from
    district update */
                                              @d_id,                  /* input
    param          */
                                              @w_id,              /* input
    param          */
                                              @li_no,             /* orderline
    number     */
                                              @li_id,             /* lineitem
    id          */
                                              @li_s_w_id,         /* lineitem
    warehouse   */
                                              "dec 31, 1889",     /* constant
    */
                                              @li_qty,            /* lineitem
    qty         */
                                              @i_price * @li_qty, /* ol_amount
    */
                                              @s_dist)            /* from
    stock           */

                /* send line-item data to client */

                       select @i_name,
                       @s_quantity,
                       b_g = case when ( (patindex("%ORIGINAL%",@i_data) > 0)
    and
                                         (patindex("%ORIGINAL%",@s_data) > 0)
    )
                            then "B" else "G" end,
                       @i_price,
                       @i_price * @li_qty

            end
            else
                   begin
```

```
            /*  no item found - triggers rollback condition */


                select "",0,"",0,0
                select @commit_flag = 0

        end
    end

/* get customer last name, discount, and credit rating */

select @c_last     = c_last,
       @c_discount = c_discount,
       @c_credit   = c_credit,
              @c_id_local = c_id
from customer holdlock
where c_id   = @c_id and
      c_w_id = @w_id and
      c_d_id = @d_id

/* insert fresh row into orders table */

insert into orders values (@o_id,
                           @d_id,
                              @w_id,
                                     @c_id_local,
                                     @o_entry_d,
                                         0,
                                         @o_ol_cnt,
                                         @o_all_local)

   /* insert corresponding row into new-order table */

   insert into new_order values (@o_id,
                                                          @d_id,
                                                          @w_id)


   /*  select warehouse tax */

   select @w_tax = w_tax
   from warehouse holdlock
   where w_id = @w_id

   if (@commit_flag = 1)
          commit transaction n
   else
          /* all that work for nuthin!!! */
          rollback transaction n

   /* return order data to client */
   select @w_tax,
          @d_tax,
             @o_id,
             @c_last,
             @c_discount,
             @c_credit,
             @o_entry_d,
             @commit_flag
end

go
```

PAYMENT.SQL

```
/*  File:       PAYMENT.SQL
*/
/*              Microsoft TPC-C Kit Ver. 3.00.000
*/
/*              Audited 08/23/96, By Francois Raab
*/
/*
*/
/*              Copyright Microsoft, 1996
*/
/*
*/
/*  Purpose:    Payment transaction for Microsoft TPC-C Benchmark Kit
*/
/*  Author:     Damien Lindauer
*/
/*              damienl@Microsoft.com
*/


use tpcc
go

if exists (select name from sysobjects where name = "tpcc_payment" )
       drop procedure tpcc_payment
go

create proc tpcc_payment @w_id           smallint,
                                              @c_w_id         smallint,
                                              @h_amount
numeric(6,2),
                                              @d_id           tinyint,
                                              @c_d_id         tinyint,
                                              @c_id           int,
                                              @c_last         char(16) =
""


as
declare  @w_street_1     char(20),
         @w_street_2     char(20),
         @w_city         char(20),
         @w_state        char(2),
         @w_zip          char(9),
         @w_name         char(10),
         @d_street_1     char(20),
         @d_street_2     char(20),
         @d_city         char(20),
         @d_state        char(2),
         @d_zip          char(9),
         @d_name         char(10),
         @c_first        char(16),
```

```
        @c_middle       char(2),
        @c_street_1     char(20),
        @c_street_2     char(20),
        @c_city         char(20),
        @c_state        char(2),
        @c_zip          char(9),
        @c_phone        char(16),
        @c_since        datetime,
        @c_credit       char(2),
        @c_credit_lim   numeric(12,2),
        @c_balance      numeric(12,2),
        @c_discount     numeric(4,4),
        @data1          char(250),
        @data2          char(250),
        @c_data_1       char(250),
        @c_data_2       char(250),
        @datetime       datetime,
        @w_ytd          numeric(12,2),
        @d_ytd          numeric(12,2),
        @cnt            smallint,
        @val            smallint,
        @screen_data  char(200),
               @d_id_local    tinyint,
               @w_id_local    smallint,
               @c_id_local    int

select @screen_data = ""

begin tran p

        /* get payment date */

        select @datetime = getdate()

        if (@c_id = 0)
        begin
               /* get customer id and info using last name */

               select @cnt = count(*)
               from customer holdlock
               where c_last = @c_last and
                     c_w_id = @c_w_id and
                     c_d_id = @c_d_id


               select @val = (@cnt + 1) / 2
               set rowcount @val

               select @c_id = c_id
               from customer holdlock
               where c_last = @c_last and
                     c_w_id = @c_w_id and
                     c_d_id = @c_d_id
               order by c_w_id, c_d_id, c_last, c_first

               set rowcount 0
        end

        /* get customer info and update balances */

        update customer set
               c_balance     = c_balance = c_balance - @h_amount,
               c_payment_cnt  = c_payment_cnt + 1,
               c_ytd_payment  = c_ytd_payment + @h_amount,
               @c_first       = c_first,
               @c_middle      = c_middle,
           @c_last        = c_last,
           @c_street_1    = c_street_1,
               @c_street_2    = c_street_2,
               @c_city        = c_city,
               @c_state       = c_state,
               @c_zip         = c_zip,
               @c_phone       = c_phone,
               @c_credit      = c_credit,
               @c_credit_lim  = c_credit_lim,
               @c_discount    = c_discount,
               @c_since       = c_since,
               @data1         = c_data_1,
               @data2         = c_data_2,
               @c_id_local    = c_id
        where c_id   = @c_id and
              c_w_id = @c_w_id and
              c_d_id = @c_d_id

        /* if customer has bad credit get some more info */

        if (@c_credit = "BC")
        begin

               /* compute new info */

               select @c_data_2 = substring(@data1,209,42) +
                                             substring(@data2, 1, 208)
               select @c_data_1 = convert(char(5),@c_id) +
                                             convert(char(4),@c_d_id) +
                                      convert(char(5),@c_w_id) +
                                      convert(char(4),@d_id) +
                                      convert(char(5),@w_id) +
                                      convert(char(19),@h_amount) +
                                      substring(@data1, 1, 208)

               /* update customer info */

               update customer set
                      c_data_1 = @c_data_1,
                          c_data_2 = @c_data_2
               where c_id   = @c_id and
                         c_w_id = @c_w_id and
                     c_d_id = @c_d_id

               select @screen_data = substring (@c_data_1,1,200)
        end



        /* get district data and update year-to-date */

        update district
               set d_ytd       = d_ytd + @h_amount,
                      @d_street_1 = d_street_1,
                      @d_street_2 = d_street_2,
                      @d_city     = d_city,
```

```
                @d_state    = d_state,
                @d_zip      = d_zip,
                @d_name     = d_name,
                @d_id_local = d_id
        where d_w_id = @w_id and
              d_id   = @d_id

/* get warehouse data and update year-to-date */

update warehouse
set w_ytd       = w_ytd + @h_amount,
       @w_street_1 = w_street_1,
    @w_street_2 = w_street_2,
    @w_city     = w_city,
    @w_state    = w_state,
    @w_zip      = w_zip,
    @w_name     = w_name,
       @w_id_local = w_id
where w_id = @w_id

/* create history record */

insert into history values  (@c_id_local,
                                                    @c_d_id,
                                                        @c_w_id,
                                                        @d_id_local,
                                                        @w_id_local,
                                                        @datetime,
                                                        @h_amount,
                                                        @w_name + "
" + @d_name)

commit tran p

/* return data to client */

select  @c_id,
                @c_last,
                @datetime,
                @w_street_1,
                @w_street_2,
                @w_city,
                @w_state,
                @w_zip,
                @d_street_1,
                @d_street_2,
                @d_city,
                @d_state,
                @d_zip,
                @c_first,
                @c_middle,
                @c_street_1,
                @c_street_2,
                @c_city,
                @c_state,
                @c_zip,
                @c_phone,
                @c_since,
                @c_credit,
                @c_credit_lim,
                @c_discount,
                @c_balance,
                @screen_data

go
```

DELIVERY.SQL

```
/*  File:       DELIVERY.SQL
*/
/*              Microsoft TPC-C Kit Ver. 3.00.000
*/
/*              Audited 08/23/96, By Francois Raab
*/
/*
*/
/*              Copyright Microsoft, 1996
*/
/*
*/
/*  Purpose:    Delivery transaction for Microsoft TPC-C Benchmark Kit
*/
/*  Author:     Damien Lindauer
*/
/*              damienl@Microsoft.com
*/


use tpcc
go

/* delivery transaction */

if exists (select name from sysobjects where name = "tpcc_delivery" )
      drop procedure tpcc_delivery
go

create proc tpcc_delivery                   @w_id           smallint,

                                                            @o_carrier_id
smallint
as

declare @d_id tinyint,
        @o_id int,
        @c_id int,
        @total numeric(12,2),
        @oid1 int,
        @oid2 int,
        @oid3 int,
        @oid4 int,
        @oid5 int,
        @oid6 int,
        @oid7 int,
        @oid8 int,
        @oid9 int,
        @oid10 int
```

```
select @d_id = 0

begin tran d

    while (@d_id < 10)
    begin

        select @d_id  = @d_id + 1,
               @total = 0,
               @o_id  = 0

        select @o_id = min(no_o_id)
        from new_order holdlock
        where no_w_id = @w_id and
                     no_d_id = @d_id

        if (@@rowcount <> 0)
        begin

            /* claim the order for this district */

            delete new_order
            where no_w_id = @w_id and
                         no_d_id = @d_id and
                             no_o_id = @o_id

            /* set carrier_id on this order (and get customer id) */

            update orders
                  set o_carrier_id = @o_carrier_id,
                @c_id        = o_c_id
            where o_w_id = @w_id and
                         o_d_id = @d_id and
                             o_id   = @o_id

            /* set date in all lineitems for this order (and sum amounts)
*/

            update order_line
                   set ol_delivery_d = getdate(),
                    @total        = @total + ol_amount
            where ol_w_id = @w_id and
                  ol_d_id = @d_id and
                             ol_o_id = @o_id

            /* accummulate lineitem amounts for this order into customer
*/

                   update customer
                      set c_balance      = c_balance + @total,
                          c_delivery_cnt = c_delivery_cnt + 1

            where c_w_id = @w_id and
                         c_d_id = @d_id and
                             c_id   = @c_id

        end

        select @oid1 = case @d_id when  1  then @o_id else @oid1 end,
               @oid2 = case @d_id when  2  then @o_id else @oid2 end,
               @oid3 = case @d_id when  3  then @o_id else @oid3 end,
               @oid4 = case @d_id when  4  then @o_id else @oid4 end,
               @oid5 = case @d_id when  5  then @o_id else @oid5 end,
               @oid6 = case @d_id when  6  then @o_id else @oid6 end,
               @oid7 = case @d_id when  7  then @o_id else @oid7 end,
               @oid8 = case @d_id when  8  then @o_id else @oid8 end,
               @oid9 = case @d_id when  9  then @o_id else @oid9 end,
               @oid10 = case @d_id when 10 then @o_id else @oid10 end

    end

commit tran d

select @oid1,
       @oid2,
       @oid3,
       @oid4,
       @oid5,
       @oid6,
       @oid7,
       @oid8,
       @oid9,
       @oid10

go
```

ORDSTAT.SQL

```
/*  File:       ORDSTAT.SQL
*/
/*              Microsoft TPC-C Kit Ver. 3.00.000
*/
/*              Audited 08/23/96, By Francois Raab
*/
/*
*/
/*              Copyright Microsoft, 1996
*/
/*
*/
/*  Purpose:    Order-Status transaction for Microsoft TPC-C Benchmark Kit
*/
/*  Author:     Damien Lindauer
*/
/*              damienl@Microsoft.com
*/


use tpcc
go

if exists ( select name from sysobjects where name = "tpcc_orderstatus" )
       drop procedure   tpcc_orderstatus
go

/*  Modified by rick vicik, 2/4/97  */
/*  Eliminated @val local variable  */

create proc tpcc_orderstatus @w_id          smallint,
```

```
                                                @d_id
        tinyint,
                                                @c_id         int,
                                                @c_last
        char(16) = ""

as

declare @c_balance            numeric(12,2),
        @c_first              char(16),
        @c_middle             char(2),
        @o_id                 int,
        @o_entry_d            datetime,
        @o_carrier_id  smallint,
        @cnt                  smallint

begin tran o

        if (@c_id = 0)
                begin
                /* get customer id and info using last name */


                select @cnt = (count(*)+1)/2
                from customer holdlock
                where c_last = @c_last and
                  c_w_id = @w_id and
                  c_d_id = @d_id
                set rowcount @cnt

                select @c_id = c_id,
                          @c_balance = c_balance,
                          @c_first   = c_first,
                          @c_last    = c_last,
                          @c_middle  = c_middle
                from customer holdlock
                where c_last = @c_last and
                  c_w_id = @w_id and
                  c_d_id = @d_id
                order by c_w_id, c_d_id, c_last, c_first

                set rowcount 0
                end

        else
                begin

                /*  get customer info if by id*/

                select @c_balance = c_balance,
                       @c_first   = c_first,
                       @c_middle  = c_middle,
                       @c_last    = c_last
                from customer holdlock
                where c_id   = @c_id and
                      c_d_id = @d_id and
                      c_w_id = @w_id

                select @cnt = @@rowcount
                end

        /* if no such customer */
        if (@cnt = 0)
        begin
                raiserror("Customer not found",18,1)
                goto custnotfound
        end

        /*  get order info */

        select @o_id = o_id,
                  @o_entry_d    = o_entry_d,
           @o_carrier_id = o_carrier_id
        from orders holdlock
    where o_w_id = @w_id and
          o_d_id = @d_id and
           o_c_id = @c_id

        /*  select order lines for the current order */

        select ol_supply_w_id,
           ol_i_id,
               ol_quantity,
               ol_amount,
               ol_delivery_d
        from order_line holdlock
        where ol_o_id = @o_id and
              ol_d_id = @d_id and
              ol_w_id = @w_id

custnotfound:

commit tran o

/*  return data to client  */

select @c_id,
          @c_last,
          @c_first,
          @c_middle,
          @o_entry_d,
          @o_carrier_id,
          @c_balance,
          @o_id

go
```

STOCKLEV.SQL

```
/*  File:       STOCKLEV.SQL
*/
/*              Microsoft TPC-C Kit Ver. 3.00.000
*/
/*              Audited 08/23/96, By Francois Raab
*/
/*
*/
```

```
/*              Copyright Microsoft, 1996
*/
/*
*/
/*  Purpose:    Stock-Level transaction for Microsoft TPC-C Benchmark Kit
*/
/*  Author:     Damien Lindauer
*/
/*              damienl@Microsoft.com
*/


use tpcc
go

/* stock-level transaction stored procedure */

if exists (select name from sysobjects where name = "tpcc_stocklevel" )
       drop procedure tpcc_stocklevel
go

/* Modified by rick vicik, 2/4/97 */
/* Eliminate 1 local variable, use derived table to eliminate duplicate
item#'s */

create proc tpcc_stocklevel   @w_id          smallint,
                              @d_id          tinyint,
                              @threshhold    smallint
as
declare @o_id int

select @o_id = d_next_o_id
from district
where d_w_id = @w_id and
         d_id   = @d_id

select count(*) from stock,
    (select distinct(ol_i_id) from order_line
     where ol_w_id    =  @w_id and
           ol_d_id    =  @d_id and
           ol_o_id between (@o_id-20) and (@o_id-1)) OL

where s_w_id     =  @w_id and
      s_i_id     =  OL.ol_i_id and
      s_quantity <  @threshhold
go
```

Loader Source

TPCCLDR.C

```
/*      FILE:          TPCCLDR.C
 *                     Microsoft TPC-C Kit Ver. 3.00.000
 *                     Audited 08/23/96, By Francois Raab
 *
 *                     Copyright Microsoft, 1996
 *
 *      PURPOSE:       Database loader for Microsoft TPC-C Benchmark Kit
 *      Author:        Damien Lindauer
 *                     damienl@Microsoft.com
 */


// Includes
#include "tpcc.h"
#include "search.h"

// Defines
#define MAXITEMS                 100000
#define CUSTOMERS_PER_DISTRICT   3000
#define DISTRICT_PER_WAREHOUSE   10
#define ORDERS_PER_DISTRICT      3000
#define MAX_CUSTOMER_THREADS     2
#define MAX_ORDER_THREADS        3
#define MAX_MAIN_THREADS         4

// Functions declarations
long NURand();
void LoadItem();
void LoadWarehouse();

void Stock();
void District();

void LoadCustomer();
void CustomerBufInit();
void CustomerBufLoad();
void LoadCustomerTable();
void LoadHistoryTable();

void LoadOrders();
void OrdersBufInit();
void OrdersBufLoad();
void LoadOrdersTable();
void LoadNewOrderTable();
void LoadOrderLineTable();
void GetPermutation();
void CheckForCommit();
void OpenConnections();

void BuildIndex();

void CurrentDate();

// Shared memory structures

typedef struct
{
    long                ol;
    long                ol_i_id;
    short               ol_supply_w_id;
    short               ol_quantity;
    double              ol_amount;
    char                ol_dist_info[DIST_INFO_LEN+1];
       // Added to insure ol_delivery_d set properly during load
       char                          ol_delivery_d[30];
} ORDER_LINE_STRUCT;

typedef struct
{
```

```
    long                o_id;
    short               o_d_id;
    short               o_w_id;
    long                o_c_id;
    short               o_carrier_id;
    short               o_ol_cnt;
    short               o_all_local;
       ORDER_LINE_STRUCT   o_ol[15];
} ORDERS_STRUCT;

typedef struct
{
    long                            c_id;
    short                           c_d_id;
    short                           c_w_id;
       char                         c_first[FIRST_NAME_LEN+1];
    char                            c_middle[MIDDLE_NAME_LEN+1];
    char                            c_last[LAST_NAME_LEN+1];
    char                            c_street_1[ADDRESS_LEN+1];
    char                            c_street_2[ADDRESS_LEN+1];
    char                            c_city[ADDRESS_LEN+1];
    char                            c_state[STATE_LEN+1];
    char                            c_zip[ZIP_LEN+1];
    char                            c_phone[PHONE_LEN+1];
    char                            c_credit[CREDIT_LEN+1];
    double                          c_credit_lim;
    double                          c_discount;
    double                          c_balance;
    double                          c_ytd_payment;
       short                        c_payment_cnt;
       short                        c_delivery_cnt;
       char                         c_data_1[C_DATA_LEN+1];
    char                            c_data_2[C_DATA_LEN+1];
       double                       h_amount;
    char                            h_data[H_DATA_LEN+1];
} CUSTOMER_STRUCT;

typedef struct
{
       char                         c_last[LAST_NAME_LEN+1];
       char                         c_first[FIRST_NAME_LEN+1];
    long                            c_id;
} CUSTOMER_SORT_STRUCT;

typedef struct
{
    long                time_start;
} LOADER_TIME_STRUCT;


// Global variables
char            errfile[20];
DBPROCESS       *i_dbproc1;
DBPROCESS       *w_dbproc1, *w_dbproc2;
DBPROCESS       *c_dbproc1, *c_dbproc2;
DBPROCESS       *o_dbproc1, *o_dbproc2, *o_dbproc3;
ORDERS_STRUCT   orders_buf[ORDERS_PER_DISTRICT];
CUSTOMER_STRUCT customer_buf[CUSTOMERS_PER_DISTRICT];
long            main_threads_completed;
long            customer_threads_completed;
long            order_threads_completed;
long            orders_rows_loaded;
long            new_order_rows_loaded;
long            order_line_rows_loaded;
long            history_rows_loaded;
long            customer_rows_loaded;
long            stock_rows_loaded;
long            district_rows_loaded;
long            item_rows_loaded;
long            warehouse_rows_loaded;
long            main_time_start;
long            main_time_end;
TPCCLDR_ARGS    *aptr, args;


//========================================================================
//
// Function name: main
//
//========================================================================

int main(int  argc, char **argv)
{

       DWORD           dwThreadID[MAX_MAIN_THREADS];
       HANDLE          hThread[MAX_MAIN_THREADS];
       FILE            *fLoader;
       char            buffer[255];
       int             main_threads_started;
       RETCODE retcode;
       LOGINREC *login;

    printf("\n*************************************************");
       printf("\n*                                               *");
       printf("\n*  Microsoft SQL Server 6.5                     *");
       printf("\n*                                               *");
       printf("\n*  TPC-C BENCHMARK KIT:  Database loader        *");
       printf("\n*  Version %s                                *",
TPCKIT_VER);
       printf("\n*                                               *");
       printf("\n*************************************************\n\n"
);

    // process command line arguments

    aptr = &args;
    GetArgsLoader(argc, argv, aptr);

       if (aptr->build_index = 0)
              printf("data load only\n");
       if (aptr->build_index = 1)
              printf("data load and index creation\n");

       // install dblib error handlers

       dbmsghandle((DBMSGHANDLE_PROC)SQLMsgHandler);
       dberrhandle((DBERRHANDLE_PROC)SQLErrHandler);

       // open connections to SQL Server
```

```
        OpenConnections();

        // open file for loader results
    fLoader = fopen(aptr->loader_res_file, "a");

    if (fLoader == NULL)
        {
                printf("Error, loader result file open failed.");
                exit(-1);
        }

    // start loading data

    sprintf(buffer,"TPC-C load started for %ld warehouses:  ", aptr-
>num_warehouses);
        if(aptr->build_index = 0)
                strcat(buffer, "data load only\n");
        if (aptr->build_index = 1)
                strcat(buffer, "data load and index creation\n");

        printf("%s",buffer);
        fprintf(fLoader,"%s",buffer);

        main_time_start = (TimeNow() / MILLI);

        // start parallel load threads

        main_threads_completed = 0;
        main_threads_started = 0;

        if ((aptr->table == NULL) || !(strcmp(aptr->table,"item")))
        {
                fprintf(fLoader, "\nStarting loader threads for: item\n");

                hThread[0] = CreateThread(NULL,
                                                                0,

(LPTHREAD_START_ROUTINE) LoadItem,
                                                                NULL,
                                                                0,

&dwThreadID[0]);

                if (hThread[0] == NULL)
                {
                        printf("Error, failed in creating creating thread =
0.\n");
                        exit(-1);
                }

                main_threads_started++;

        }

        if ((aptr->table == NULL) || !(strcmp(aptr->table,"warehouse")))
        {
                fprintf(fLoader, "Starting loader threads for:
warehouse\n");

                hThread[1] = CreateThread(NULL,
                                                                0,

(LPTHREAD_START_ROUTINE) LoadWarehouse,
                                                                NULL,
                                                                0,

&dwThreadID[1]);

                if (hThread[1] == NULL)
                {
                        printf("Error, failed in creating creating thread =
1.\n");
                        exit(-1);
                }

                main_threads_started++;
        }

    if ((aptr->table == NULL) || !(strcmp(aptr->table,"customer")))
        {
                fprintf(fLoader, "Starting loader threads for:
customer\n");

                hThread[2] = CreateThread(NULL,
                                                                0,

(LPTHREAD_START_ROUTINE) LoadCustomer,
                                                                NULL,
                                                                0,

&dwThreadID[2]);

                if (hThread[2] == NULL)
                {
                        printf("Error, failed in creating creating main
thread = 2.\n");
                        exit(-1);
                }

                main_threads_started++;
        }

    if ((aptr->table == NULL) || !(strcmp(aptr->table,"orders")))
        {
                fprintf(fLoader, "Starting loader threads for: orders\n");

                hThread[3] = CreateThread(NULL,
                                                                0,

(LPTHREAD_START_ROUTINE) LoadOrders,
                                                                NULL,
                                                                0,

&dwThreadID[3]);

                if (hThread[3] == NULL)
                {
                        printf("Error, failed in creating creating main
thread = 3.\n");
                        exit(-1);
                }
```

```
        main_threads_started++;

        }

        while (main_threads_completed != main_threads_started)
                Sleep(1000L);

    main_time_end = (TimeNow() / MILLI);

    sprintf(buffer,"\nTPC-C load completed successfully in %ld
minutes.\n",
                        (main_time_end - main_time_start)/60);

        printf("%s",buffer);
        fprintf(fLoader, "%s", buffer);

        fclose(fLoader);

        dbexit();

    exit(0);
}


//========================================================================
//
// Function name: LoadItem
//
//========================================================================

void LoadItem()
{
    long   i_id;
       long   i_im_id;
    char   i_name[I_NAME_LEN+1];
    double i_price;
    char   i_data[I_DATA_LEN+1];
       char   name[20];
       long   time_start;

       printf("\nLoading item table...\n");

       // Seed with unique number
       seed(1);

       InitString(i_name, I_NAME_LEN+1);
       InitString(i_data, I_DATA_LEN+1);

       sprintf(name, "%s..%s", aptr->database, "item");
       bcp_init(i_dbproc1, name, NULL, "logs\\item.err", DB_IN);

       bcp_bind(i_dbproc1, (BYTE *) &i_id,      0, -1,         NULL, 0,
0, 1);
       bcp_bind(i_dbproc1, (BYTE *) &i_im_id,   0, -1,         NULL, 0,
0, 2);
       bcp_bind(i_dbproc1, (BYTE *) i_name,     0, I_NAME_LEN, NULL, 0,
0, 3);
       bcp_bind(i_dbproc1, (BYTE *) &i_price,   0, -1,         NULL, 0,
SQLFLT8, 4);
       bcp_bind(i_dbproc1, (BYTE *) i_data,     0, I_DATA_LEN, NULL, 0,
0, 5);

       time_start = (TimeNow() / MILLI);

       item_rows_loaded = 0;

       for (i_id = 1; i_id <= MAXITEMS; i_id++)
       {
              i_im_id = RandomNumber(1L, 10000L);

              MakeAlphaString(14, 24, I_NAME_LEN, i_name);

              i_price = ((float) RandomNumber(100L, 10000L))/100.0;

              MakeOriginalAlphaString(26, 50, I_DATA_LEN, i_data, 10);

              if (!bcp_sendrow(i_dbproc1))
                     printf("Error, LoadItem() failed calling
bcp_sendrow().  Check error file.\n");
              item_rows_loaded++;
              CheckForCommit(i_dbproc1, item_rows_loaded, "item",
&time_start);
       }

       bcp_done(i_dbproc1);
       dbclose(i_dbproc1);

       printf("Finished loading item table.\n");

       if (aptr->build_index == 1)
              BuildIndex("idxitmcl");

       InterlockedIncrement(&main_threads_completed);
}


//========================================================================
//
// Function   : LoadWarehouse
//
// Loads WAREHOUSE table and loads Stock and District as Warehouses are
created
//
//========================================================================

void LoadWarehouse()
{
    short  w_id;
    char   w_name[W_NAME_LEN+1];
    char   w_street_1[ADDRESS_LEN+1];
    char   w_street_2[ADDRESS_LEN+1];
    char   w_city[ADDRESS_LEN+1];
    char   w_state[STATE_LEN+1];
    char   w_zip[ZIP_LEN+1];
    double w_tax;
    double w_ytd;
       char   name[20];
       long   time_start;
```

```
        printf("\nLoading warehouse table...\n");

        // Seed with unique number
        seed(2);

        InitString(w_name, W_NAME_LEN+1);
        InitAddress(w_street_1, w_street_2, w_city, w_state, w_zip);

        sprintf(name, "%s..%s", aptr->database, "warehouse");
        bcp_init(w_dbproc1, name, NULL, "logs\\whouse.err", DB_IN);

        bcp_bind(w_dbproc1, (BYTE *) &w_id,         0, -1,           NULL,
0, 0, 1);
        bcp_bind(w_dbproc1, (BYTE *) w_name,        0, W_NAME_LEN,   NULL,
0, 0, 2);
        bcp_bind(w_dbproc1, (BYTE *) w_street_1,    0, ADDRESS_LEN,  NULL,
0, 0, 3);
        bcp_bind(w_dbproc1, (BYTE *) w_street_2,    0, ADDRESS_LEN,  NULL,
0, 0, 4);
        bcp_bind(w_dbproc1, (BYTE *) w_city,        0, ADDRESS_LEN,  NULL,
0, 0, 5);
        bcp_bind(w_dbproc1, (BYTE *) w_state,       0, STATE_LEN,    NULL,
0, 0, 6);
        bcp_bind(w_dbproc1, (BYTE *) w_zip,         0, ZIP_LEN,      NULL,
0, 0, 7);
        bcp_bind(w_dbproc1, (BYTE *) &w_tax,        0, -1,           NULL,
0, SQLFLT8, 8);
        bcp_bind(w_dbproc1, (BYTE *) &w_ytd,        0, -1,           NULL,
0, SQLFLT8, 9);

        time_start = (TimeNow() / MILLI);

        warehouse_rows_loaded = 0;

        for (w_id = aptr->starting_warehouse; w_id < aptr-
>num_warehouses+1; w_id++)
        {

                MakeAlphaString(6,10, W_NAME_LEN, w_name);

                MakeAddress(w_street_1, w_street_2, w_city, w_state,
w_zip);

                w_tax = ((float) RandomNumber(0L,2000L))/10000.00;

                w_ytd = 300000.00;

                if (!bcp_sendrow(w_dbproc1))
                printf("Error, LoadWarehouse() failed calling
bcp_sendrow().  Check error file.\n");
                warehouse_rows_loaded++;
                CheckForCommit(i_dbproc1, warehouse_rows_loaded,
"warehouse", &time_start);
        }

        bcp_done(w_dbproc1);
        dbclose(w_dbproc1);

        printf("Finished loading warehouse table.\n");

        if (aptr->build_index == 1)
                BuildIndex("idxwarcl");

        stock_rows_loaded = 0;
        district_rows_loaded = 0;

        District(w_id);
        Stock(w_id);

        InterlockedIncrement(&main_threads_completed);
}


//============================================================================
//
// Function   : District
//
//============================================================================

void District()
{
    short  d_id;
    short  d_w_id;
    char   d_name[D_NAME_LEN+1];
    char   d_street_1[ADDRESS_LEN+1];
    char   d_street_2[ADDRESS_LEN+1];
    char   d_city[ADDRESS_LEN+1];
    char   d_state[STATE_LEN+1];
    char   d_zip[ZIP_LEN+1];
    double d_tax;
    double d_ytd;
       char   name[20];
    long   d_next_o_id;
    int    rc;
       long   time_start;
       int       w_id;

       for (w_id = aptr->starting_warehouse; w_id < aptr-
>num_warehouses+1; w_id++)
       {

               printf("...Loading district table: w_id = %ld\n", w_id);

               // Seed with unique number
               seed(4);

               InitString(d_name, D_NAME_LEN+1);

               InitAddress(d_street_1, d_street_2, d_city, d_state,
d_zip);

               sprintf(name, "%s..%s", aptr->database, "district");
               rc = bcp_init(w_dbproc2, name, NULL, "logs\\district.err",
DB_IN);

               bcp_bind(w_dbproc2, (BYTE *) &d_id,          0, -1,
NULL, 0, 0, 1);
               bcp_bind(w_dbproc2, (BYTE *) &d_w_id,        0, -1,
NULL, 0, 0, 2);
               bcp_bind(w_dbproc2, (BYTE *) d_name,         0, D_NAME_LEN,
NULL, 0, 0, 3);
```

```
               bcp_bind(w_dbproc2, (BYTE *) d_street_1,     0,
ADDRESS_LEN,   NULL, 0, 0, 4);
               bcp_bind(w_dbproc2, (BYTE *) d_street_2,     0,
ADDRESS_LEN,   NULL, 0, 0, 5);
               bcp_bind(w_dbproc2, (BYTE *) d_city,         0,
ADDRESS_LEN,   NULL, 0, 0, 6);
               bcp_bind(w_dbproc2, (BYTE *) d_state,        0, STATE_LEN,
NULL, 0, 0, 7);
               bcp_bind(w_dbproc2, (BYTE *) d_zip,          0, ZIP_LEN,
NULL, 0, 0, 8);
               bcp_bind(w_dbproc2, (BYTE *) &d_tax,         0, -1,
NULL, 0, SQLFLT8, 9);
               bcp_bind(w_dbproc2, (BYTE *) &d_ytd,         0, -1,
NULL, 0, SQLFLT8, 10);
               bcp_bind(w_dbproc2, (BYTE *) &d_next_o_id,   0, -1,
NULL, 0, 0, 11);

               d_w_id = w_id;

               d_ytd  = 30000.0;

               d_next_o_id = 3001L;

               time_start = (TimeNow() / MILLI);

               for (d_id = 1; d_id <= DISTRICT_PER_WAREHOUSE; d_id++)
               {
                      MakeAlphaString(6,10,D_NAME_LEN, d_name);

                      MakeAddress(d_street_1, d_street_2, d_city,
d_state, d_zip);

                      d_tax = ((float) RandomNumber(0L,2000L))/10000.00;

                      if (!bcp_sendrow(w_dbproc2))
                             printf("Error, District() failed calling
bcp_sendrow().  Check error file.\n");
                      district_rows_loaded++;
                      CheckForCommit(w_dbproc2, district_rows_loaded,
"district", &time_start);
               }

               rc = bcp_done(w_dbproc2);
       }

       printf("Finished loading district table.\n");

       if (aptr->build_index == 1)
               BuildIndex("idxdiscl");

    return;
}


//=============================================================================
//
// Function   : Stock
//
//=============================================================================
```

```
void Stock()
{
    long  s_i_id;
    short s_w_id;
    short s_quantity;
    char  s_dist_01[S_DIST_LEN+1];
    char  s_dist_02[S_DIST_LEN+1];
    char  s_dist_03[S_DIST_LEN+1];
    char  s_dist_04[S_DIST_LEN+1];
    char  s_dist_05[S_DIST_LEN+1];
    char  s_dist_06[S_DIST_LEN+1];
    char  s_dist_07[S_DIST_LEN+1];
    char  s_dist_08[S_DIST_LEN+1];
    char  s_dist_09[S_DIST_LEN+1];
    char  s_dist_10[S_DIST_LEN+1];
    long  s_ytd;
    short s_order_cnt;
    short s_remote_cnt;
    char  s_data[S_DATA_LEN+1];
    short i;
    short len;
    int   rc;
       char  name[20];
       long  time_start;

       // Seed with unique number
       seed(3);

       sprintf(name, "%s..%s", aptr->database, "stock");
       rc = bcp_init(w_dbproc2, name, NULL, "logs\\stock.err", DB_IN);

       bcp_bind(w_dbproc2, (BYTE *) &s_i_id,       0, -1,         NULL,
0, 0, 1);
       bcp_bind(w_dbproc2, (BYTE *) &s_w_id,       0, -1,         NULL,
0, 0, 2);
       bcp_bind(w_dbproc2, (BYTE *) &s_quantity,   0, -1,         NULL,
0, 0, 3);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_01,     0, S_DIST_LEN, NULL,
0, 0, 4);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_02,     0, S_DIST_LEN, NULL,
0, 0, 5);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_03,     0, S_DIST_LEN, NULL,
0, 0, 6);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_04,     0, S_DIST_LEN, NULL,
0, 0, 7);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_05,     0, S_DIST_LEN, NULL,
0, 0, 8);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_06,     0, S_DIST_LEN, NULL,
0, 0, 9);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_07,     0, S_DIST_LEN, NULL,
0, 0, 10);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_08,     0, S_DIST_LEN, NULL,
0, 0, 11);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_09,     0, S_DIST_LEN, NULL,
0, 0, 12);
       bcp_bind(w_dbproc2, (BYTE *) s_dist_10,     0, S_DIST_LEN, NULL,
0, 0, 13);
       bcp_bind(w_dbproc2, (BYTE *) &s_ytd,        0, -1,         NULL,
0, 0, 14);
```

```
        bcp_bind(w_dbproc2, (BYTE *) &s_order_cnt,  0, -1,         NULL,
0, 0, 15);
        bcp_bind(w_dbproc2, (BYTE *) &s_remote_cnt, 0, -1,         NULL,
0, 0, 16);
        bcp_bind(w_dbproc2, (BYTE *) s_data,        0, S_DATA_LEN,   NULL,
0, 0, 17);

        s_ytd = s_order_cnt = s_remote_cnt = 0;

        time_start = (TimeNow() / MILLI);

        printf("...Loading stock table\n");

        for (s_i_id=1; s_i_id <= MAXITEMS; s_i_id++)
        {

                for (s_w_id = aptr->starting_warehouse; s_w_id < aptr-
>num_warehouses+1; s_w_id++)
                {

                        s_quantity = RandomNumber(10L,100L);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_01);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_02);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_03);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_04);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_05);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_06);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_07);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_08);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_09);
                        len = MakeAlphaString(24,24,S_DIST_LEN, s_dist_10);

                        len = MakeOriginalAlphaString(26,50, S_DATA_LEN,
s_data,10);

                        if (!bcp_sendrow(w_dbproc2))
                        printf("Error, Stock() failed calling
bcp_sendrow().  Check error file.\n");
                        stock_rows_loaded++;
                        CheckForCommit(w_dbproc2, stock_rows_loaded,
"stock", &time_start);

                }
        }

        bcp_done(w_dbproc2);
        dbclose(w_dbproc2);

        printf("Finished loading stock table.\n");

        if (aptr->build_index == 1)
                BuildIndex("idxstkcl");

        return;
}


//============================================================================
//
// Function   : LoadCustomer
//
```

```
//============================================================================

void LoadCustomer()
{
        LOADER_TIME_STRUCT      customer_time_start;
        LOADER_TIME_STRUCT      history_time_start;
        short                           w_id;
    short                       d_id;
        DWORD                           dwThreadID[MAX_CUSTOMER_THREADS];
        HANDLE                          hThread[MAX_CUSTOMER_THREADS];
        char                            name[20];
        char                                    buf[250];

        printf("\nLoading customer and history tables...\n");

        // Seed with unique number
        seed(5);

        // Initialize bulk copy
        sprintf(name, "%s..%s", aptr->database, "customer");
        bcp_init(c_dbproc1, name, NULL, "logs\\customer.err", DB_IN);

        sprintf(name, "%s..%s", aptr->database, "history");
        bcp_init(c_dbproc2, name, NULL, "logs\\history.err", DB_IN);

        customer_rows_loaded    = 0;
        history_rows_loaded     = 0;

        CustomerBufInit();

        customer_time_start.time_start = (TimeNow() / MILLI);
        history_time_start.time_start = (TimeNow() / MILLI);

        for (w_id = aptr->starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
        {
                for (d_id = 1L; d_id <= DISTRICT_PER_WAREHOUSE; d_id++)
                {

                        CustomerBufLoad(d_id, w_id);

                        // Start parallel loading threads here...

                        customer_threads_completed=0;

                        // Start customer table thread

                        printf("...Loading customer table for: d_id = %d,
w_id = %d\n", d_id, w_id);

                        hThread[0] = CreateThread(NULL,
                                                                                0,

(LPTHREAD_START_ROUTINE) LoadCustomerTable,

&customer_time_start,
                                                                                0,

&dwThreadID[0]);

                        if (hThread[0] == NULL)
```

```
                        {
                                printf("Error, failed in creating creating
thread = 0.\n");
                                exit(-1);
                        }

                        // Start History table thread

                        printf("...Loading history table for: d_id = %d,
w_id = %d\n", d_id, w_id);

                        hThread[1] = CreateThread(NULL,
                                                                        0,

(LPTHREAD_START_ROUTINE) LoadHistoryTable,

&history_time_start,
                                                                        0,

&dwThreadID[1]);

                        if (hThread[1] == NULL)
                        {
                                printf("Error, failed in creating creating
thread = 1.\n");
                                exit(-1);
                        }

                        while (customer_threads_completed != 2)
                                Sleep(1000L);
                }
        }

        // flush the bulk connection
        bcp_done(c_dbproc1);
        bcp_done(c_dbproc2);

    sprintf(buf,"update customer set c_first = 'C_LOAD = %d' where c_id =
1 and c_w_id = 1 and c_d_id = 1",LOADER_NURAND_C);
        dbcmd(c_dbproc1, buf);
        dbsqlexec(c_dbproc1);
        while (dbresults(c_dbproc1) != NO_MORE_RESULTS);

        dbclose(c_dbproc1);
        dbclose(c_dbproc2);

        printf("Finished loading customer table.\n");

        if (aptr->build_index == 1)
                BuildIndex("idxcuscl");

        if (aptr->build_index == 1)
                BuildIndex("idxcusnc");

        InterlockedIncrement(&main_threads_completed);

    return;

}
```

```
//============================================================================
//
// Function   : CustomerBufInit
//
//============================================================================

void CustomerBufInit()
{
        int     i;

        for (i=0;i<CUSTOMERS_PER_DISTRICT;i++)
        {
                customer_buf[i].c_id = 0;
                customer_buf[i].c_d_id = 0;
                customer_buf[i].c_w_id = 0;

                strcpy(customer_buf[i].c_first,"");
                strcpy(customer_buf[i].c_middle,"");
                strcpy(customer_buf[i].c_last,"");
                strcpy(customer_buf[i].c_street_1,"");
                strcpy(customer_buf[i].c_street_2,"");
                strcpy(customer_buf[i].c_city,"");
                strcpy(customer_buf[i].c_state,"");
                strcpy(customer_buf[i].c_zip,"");
                strcpy(customer_buf[i].c_phone,"");
                strcpy(customer_buf[i].c_credit,"");

                customer_buf[i].c_credit_lim = 0;
                customer_buf[i].c_discount = (float) 0;
                customer_buf[i].c_balance = 0;
                customer_buf[i].c_ytd_payment = 0;
                customer_buf[i].c_payment_cnt = 0;
                customer_buf[i].c_delivery_cnt = 0;

                strcpy(customer_buf[i].c_data_1,"");
                strcpy(customer_buf[i].c_data_2,"");

                customer_buf[i].h_amount = 0;

                strcpy(customer_buf[i].h_data,"");

        }


}


//============================================================================
//
// Function   : CustomerBufLoad
//
// Fills shared buffer for HISTORY and CUSTOMER
//============================================================================

void CustomerBufLoad(int d_id, int w_id)
{
        long                                    i;
    CUSTOMER_SORT_STRUCT    c[CUSTOMERS_PER_DISTRICT];
```

```
	for (i=0;i<CUSTOMERS_PER_DISTRICT;i++)
	{
		if (i < 1000)
			LastName(i, c[i].c_last);
		else
			LastName(NURand(255,0,999,LOADER_NURAND_C),
c[i].c_last);

		MakeAlphaString(8,16,FIRST_NAME_LEN, c[i].c_first);

		c[i].c_id = i+1;

	}

	printf("...Loading customer buffer for: d_id = %d, w_id = %d\n",
			d_id, w_id);

	for (i=0;i<CUSTOMERS_PER_DISTRICT;i++)
	{

		customer_buf[i].c_d_id = d_id;
		customer_buf[i].c_w_id = w_id;
		customer_buf[i].h_amount = 10.0;
		customer_buf[i].c_ytd_payment  = 10.0;
		customer_buf[i].c_payment_cnt  = 1;
		customer_buf[i].c_delivery_cnt = 0;

		// Generate CUSTOMER and HISTORY data

		customer_buf[i].c_id = c[i].c_id;

		strcpy(customer_buf[i].c_first, c[i].c_first);
		strcpy(customer_buf[i].c_last, c[i].c_last);

		customer_buf[i].c_middle[0] = 'O';
		customer_buf[i].c_middle[1] = 'E';

		MakeAddress(customer_buf[i].c_street_1,
				customer_buf[i].c_street_2,
				customer_buf[i].c_city,
				customer_buf[i].c_state,
				customer_buf[i].c_zip);

		MakeNumberString(16, 16, PHONE_LEN,
customer_buf[i].c_phone);

		if (RandomNumber(1L, 100L) > 10)
			customer_buf[i].c_credit[0] = 'G';
		else
			customer_buf[i].c_credit[0] = 'B';
		customer_buf[i].c_credit[1] = 'C';

		customer_buf[i].c_credit_lim = 50000.0;
		customer_buf[i].c_discount = ((float) RandomNumber(0L,
5000L)) / 10000.0;
		customer_buf[i].c_balance = -10.0;

		MakeAlphaString(250, 250, C_DATA_LEN,
customer_buf[i].c_data_1);

		MakeAlphaString(50, 250, C_DATA_LEN,
customer_buf[i].c_data_2);

		// Generate HISTORY data
		MakeAlphaString(12, 24, H_DATA_LEN,
customer_buf[i].h_data);

	}
}

//=============================================================================
//
// Function   : LoadCustomerTable
//
//=============================================================================

void LoadCustomerTable(LOADER_TIME_STRUCT *customer_time_start)
{
	int		i;
    long	c_id;
    short	c_d_id;
    short	c_w_id;
    char	c_first[FIRST_NAME_LEN+1];
    char	c_middle[MIDDLE_NAME_LEN+1];
    char	c_last[LAST_NAME_LEN+1];
    char	c_street_1[ADDRESS_LEN+1];
    char	c_street_2[ADDRESS_LEN+1];
    char	c_city[ADDRESS_LEN+1];
    char	c_state[STATE_LEN+1];
    char	c_zip[ZIP_LEN+1];
    char	c_phone[PHONE_LEN+1];
    char	c_credit[CREDIT_LEN+1];
    double	c_credit_lim;
    double	c_discount;
    double	c_balance;
    double	c_ytd_payment;
    short	c_payment_cnt;
    short	c_delivery_cnt;
    char	c_data_1[C_DATA_LEN+1];
    char	c_data_2[C_DATA_LEN+1];
	char	name[20];
	char	c_since[50];

    bcp_bind(c_dbproc1, (BYTE *) &c_id,		0, -1,
NULL,0,0, 1);
    bcp_bind(c_dbproc1, (BYTE *) &c_d_id,	0, -1,
NULL,0,0, 2);
    bcp_bind(c_dbproc1, (BYTE *) &c_w_id,	0, -1,
NULL,0,0, 3);
    bcp_bind(c_dbproc1, (BYTE *) c_first,	0, FIRST_NAME_LEN,
NULL,0,0, 4);
    bcp_bind(c_dbproc1, (BYTE *) c_middle,	0,
MIDDLE_NAME_LEN,NULL,0,0, 5);
    bcp_bind(c_dbproc1, (BYTE *) c_last,	0, LAST_NAME_LEN,
NULL,0,0, 6);
    bcp_bind(c_dbproc1, (BYTE *) c_street_1,	0, ADDRESS_LEN,
NULL,0,0, 7);
    bcp_bind(c_dbproc1, (BYTE *) c_street_2,	0, ADDRESS_LEN,
NULL,0,0, 8);
    bcp_bind(c_dbproc1, (BYTE *) c_city,	0, ADDRESS_LEN,
NULL,0,0, 9);
```

```
    bcp_bind(c_dbproc1, (BYTE *) c_state,        0, STATE_LEN,
NULL,0,0,10);
    bcp_bind(c_dbproc1, (BYTE *) c_zip,          0, ZIP_LEN,
NULL,0,0,11);
    bcp_bind(c_dbproc1, (BYTE *) c_phone,        0, PHONE_LEN,
NULL,0,0,12);
       bcp_bind(c_dbproc1, (BYTE *) c_since,        0, 50,
NULL,0,SQLCHAR,13);
    bcp_bind(c_dbproc1, (BYTE *) c_credit,       0, CREDIT_LEN,
NULL,0,0,14);
    bcp_bind(c_dbproc1, (BYTE *) &c_credit_lim,  0, -1,
NULL,0,SQLFLT8,15);
    bcp_bind(c_dbproc1, (BYTE *) &c_discount,    0, -1,
NULL,0,SQLFLT8,16);
    bcp_bind(c_dbproc1, (BYTE *) &c_balance,     0, -1,
NULL,0,SQLFLT8,17);
    bcp_bind(c_dbproc1, (BYTE *) &c_ytd_payment, 0, -1,
NULL,0,SQLFLT8,18);
    bcp_bind(c_dbproc1, (BYTE *) &c_payment_cnt, 0, -1,
NULL,0,0,19);
    bcp_bind(c_dbproc1, (BYTE *) &c_delivery_cnt,0, -1,
NULL,0,0,20);
    bcp_bind(c_dbproc1, (BYTE *) c_data_1,       0, C_DATA_LEN,
NULL,0,0,21);
    bcp_bind(c_dbproc1, (BYTE *) c_data_2,       0, C_DATA_LEN,
NULL,0,0,22);

    for (i = 0; i < CUSTOMERS_PER_DISTRICT; i++)
    {
               c_id = customer_buf[i].c_id;
               c_d_id = customer_buf[i].c_d_id;
               c_w_id = customer_buf[i].c_w_id;

               strcpy(c_first, customer_buf[i].c_first);
               strcpy(c_middle, customer_buf[i].c_middle);
               strcpy(c_last, customer_buf[i].c_last);
               strcpy(c_street_1, customer_buf[i].c_street_1);
               strcpy(c_street_2, customer_buf[i].c_street_2);
               strcpy(c_city, customer_buf[i].c_city);
               strcpy(c_state, customer_buf[i].c_state);
               strcpy(c_zip, customer_buf[i].c_zip);
               strcpy(c_phone, customer_buf[i].c_phone);
               strcpy(c_credit, customer_buf[i].c_credit);

               CurrentDate(&c_since);

               c_credit_lim = customer_buf[i].c_credit_lim;
               c_discount = customer_buf[i].c_discount;
               c_balance = customer_buf[i].c_balance;
               c_ytd_payment = customer_buf[i].c_ytd_payment;
               c_payment_cnt = customer_buf[i].c_payment_cnt;
               c_delivery_cnt = customer_buf[i].c_delivery_cnt;

               strcpy(c_data_1, customer_buf[i].c_data_1);
               strcpy(c_data_2, customer_buf[i].c_data_2);

               // Send data to server
               if (!bcp_sendrow(c_dbproc1))
               printf("Error, LoadCustomerTable() failed calling
bcp_sendrow().  Check error file.\n");
               customer_rows_loaded++;
               CheckForCommit(c_dbproc1, customer_rows_loaded, "customer",
&customer_time_start->time_start);
    }

        InterlockedIncrement(&customer_threads_completed);

}


//============================================================================
//
// Function   : LoadHistoryTable
//
//============================================================================

void LoadHistoryTable(LOADER_TIME_STRUCT *history_time_start)
{
        int         i;
    long        c_id;
    short       c_d_id;
    short       c_w_id;
        double          h_amount;
    char        h_data[H_DATA_LEN+1];
        char            h_date[50];

    bcp_bind(c_dbproc2, (BYTE *) &c_id,         0, -1,           NULL, 0,
0, 1);
    bcp_bind(c_dbproc2, (BYTE *) &c_d_id,       0, -1,           NULL, 0,
0, 2);
    bcp_bind(c_dbproc2, (BYTE *) &c_w_id,       0, -1,           NULL, 0,
0, 3);
    bcp_bind(c_dbproc2, (BYTE *) &c_d_id,       0, -1,           NULL, 0,
0, 4);
    bcp_bind(c_dbproc2, (BYTE *) &c_w_id,       0, -1,           NULL, 0,
0, 5);
    bcp_bind(c_dbproc2, (BYTE *) h_date,        0, 50,           NULL, 0,
SQLCHAR, 6);
    bcp_bind(c_dbproc2, (BYTE *) &h_amount,     0, -1,           NULL, 0,
SQLFLT8, 7);
    bcp_bind(c_dbproc2, (BYTE *) h_data,        0, H_DATA_LEN,   NULL, 0,
0, 8);

    for (i = 0; i < CUSTOMERS_PER_DISTRICT; i++)
    {
               c_id = customer_buf[i].c_id;
               c_d_id = customer_buf[i].c_d_id;
               c_w_id = customer_buf[i].c_w_id;
               h_amount = customer_buf[i].h_amount;
               strcpy(h_data, customer_buf[i].h_data);
               CurrentDate(&h_date);

               // send to server
               if (!bcp_sendrow(c_dbproc2))
               printf("Error, LoadHistoryTable() failed calling
bcp_sendrow().  Check error file.\n");
               history_rows_loaded++;
               CheckForCommit(c_dbproc2, history_rows_loaded, "history",
&history_time_start->time_start);
    }
```

```
        InterlockedIncrement(&customer_threads_completed);

}


//============================================================-
===============
//
// Function   : LoadOrders
//
//============================================================================
====

void LoadOrders()
{
        LOADER_TIME_STRUCT      orders_time_start;
        LOADER_TIME_STRUCT      new_order_time_start;
        LOADER_TIME_STRUCT      order_line_time_start;
        short                   w_id;
    short                   d_id;
        DWORD                   dwThreadID[MAX_ORDER_THREADS];
        HANDLE                  hThread[MAX_ORDER_THREADS];
        char                    name[20];


        printf("\nLoading orders...\n");

        // seed with unique number
        seed(6);

        // initialize bulk copy
        sprintf(name, "%s..%s", aptr->database, "orders");
        bcp_init(o_dbproc1, name, NULL, "logs\\orders.err", DB_IN);

        sprintf(name, "%s..%s", aptr->database, "new_order");
        bcp_init(o_dbproc2, name, NULL, "logs\\neword.err", DB_IN);

        sprintf(name, "%s..%s", aptr->database, "order_line");
        bcp_init(o_dbproc3, name, NULL, "logs\\ordline.err", DB_IN);

        orders_rows_loaded      = 0;
        new_order_rows_loaded   = 0;
        order_line_rows_loaded  = 0;

        OrdersBufInit();

        orders_time_start.time_start = (TimeNow() / MILLI);
        new_order_time_start.time_start = (TimeNow() / MILLI);
        order_line_time_start.time_start = (TimeNow() / MILLI);

        for (w_id = aptr->starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
        {
                for (d_id = 1L; d_id <= DISTRICT_PER_WAREHOUSE; d_id++)
                {

                        OrdersBufLoad(d_id, w_id);

                        // start parallel loading threads here...

                        order_threads_completed=0;

                        // start Orders table thread

                        printf("...Loading Order Table for: d_id = %d, w_id
= %d\n", d_id, w_id);

                        hThread[0] = CreateThread(NULL,
                                                                        0,

(LPTHREAD_START_ROUTINE) LoadOrdersTable,

&orders_time_start,
                                                                        0,

&dwThreadID[0]);

                        if (hThread[0] == NULL)
                        {
                                printf("Error, failed in creating creating
thread = 0.\n");
                                exit(-1);
                        }

                        // start NewOrder table thread

                        printf("...Loading New-Order Table for: d_id = %d,
w_id = %d\n", d_id, w_id);

                        hThread[1] = CreateThread(NULL,
                                                                        0,

(LPTHREAD_START_ROUTINE) LoadNewOrderTable,

&new_order_time_start,
                                                                        0,

&dwThreadID[1]);

                        if (hThread[1] == NULL)
                        {
                                printf("Error, failed in creating creating
thread = 1.\n");
                                exit(-1);
                        }

                        // start Order-Line table thread

                        printf("...Loading Order-Line Table for: d_id = %d,
w_id = %d\n", d_id, w_id);

                        hThread[2] = CreateThread(NULL,
                                                                        0,

(LPTHREAD_START_ROUTINE) LoadOrderLineTable,

&order_line_time_start,
                                                                        0,

&dwThreadID[2]);
```

```
                        if (hThread[2] == NULL)
                        {
                                printf("Error, failed in creating creating
thread = 2.\n");
                                exit(-1);
                        }

                        while (order_threads_completed != 3)
                                Sleep(1000L);

                }
        }

        printf("Finished loading orders.\n");

        InterlockedIncrement(&main_threads_completed);

    return;
}


//============================================================================
//
// Function   : OrdersBufInit
//
// Clears shared buffer for ORDERS, NEWORDER, and ORDERLINE
//
//============================================================================

void OrdersBufInit()
{
        int     i;
        int             j;

        for (i=0;i<ORDERS_PER_DISTRICT;i++)
        {
                orders_buf[i].o_id = 0;
                orders_buf[i].o_d_id = 0;
                orders_buf[i].o_w_id = 0;
                orders_buf[i].o_c_id = 0;
                orders_buf[i].o_carrier_id = 0;
                orders_buf[i].o_ol_cnt = 0;
                orders_buf[i].o_all_local = 0;

                for (j=0;j<=14;j++)
                {
                        orders_buf[i].o_ol[j].ol = 0;
                        orders_buf[i].o_ol[j].ol_i_id = 0;
                        orders_buf[i].o_ol[j].ol_supply_w_id = 0;
                        orders_buf[i].o_ol[j].ol_quantity = 0;
                        orders_buf[i].o_ol[j].ol_amount = 0;
                        strcpy(orders_buf[i].o_ol[j].ol_dist_info,"");
                }
        }


}


//============================================================================
//
// Function   : OrdersBufLoad
//
// Fills shared buffer for ORDERS, NEWORDER, and ORDERLINE
//
//============================================================================

void OrdersBufLoad(int d_id, int w_id)
{

        int     cust[ORDERS_PER_DIST+1];
        long    o_id;
        short   ol;

        printf("...Loading Order Buffer for: d_id = %d, w_id = %d\n",
                        d_id, w_id);

        GetPermutation(cust, ORDERS_PER_DIST);

        for (o_id=0;o_id<ORDERS_PER_DISTRICT;o_id++)
        {

                // Generate ORDER and NEW-ORDER data

                orders_buf[o_id].o_d_id = d_id;
                orders_buf[o_id].o_w_id = w_id;
                orders_buf[o_id].o_id = o_id+1;
                orders_buf[o_id].o_c_id = cust[o_id+1];
                orders_buf[o_id].o_ol_cnt = RandomNumber(5L, 15L);

                if (o_id < 2100)
                {
                        orders_buf[o_id].o_carrier_id = RandomNumber(1L,
10L);
                        orders_buf[o_id].o_all_local  = 1;
                }
                else
                {
                        orders_buf[o_id].o_carrier_id = 0;
                        orders_buf[o_id].o_all_local  = 1;
                }

                for (ol=0;ol<orders_buf[o_id].o_ol_cnt;ol++)
                {

                        orders_buf[o_id].o_ol[ol].ol = ol+1;
                        orders_buf[o_id].o_ol[ol].ol_i_id =
RandomNumber(1L, MAXITEMS);
                        orders_buf[o_id].o_ol[ol].ol_supply_w_id = w_id;
                        orders_buf[o_id].o_ol[ol].ol_quantity = 5;
                        MakeAlphaString(24, 24, OL_DIST_INFO_LEN,
&orders_buf[o_id].o_ol[ol].ol_dist_info);

                        // Generate ORDER-LINE data
                        if (o_id < 2100)
                        {
                                orders_buf[o_id].o_ol[ol].ol_amount = 0;
                                // Added to insure ol_delivery_d set
properly during load
```

```
        CurrentDate(&orders_buf[o_id].o_ol[ol].ol_delivery_d);
                        }
                        else
                        {
                                orders_buf[o_id].o_ol[ol].ol_amount =
RandomNumber(1,999999)/100.0;
                                // Added to insure ol_delivery_d set
properly during load

        strcpy(orders_buf[o_id].o_ol[ol].ol_delivery_d,"Dec 31, 1889");
                        }
                }
        }
}


//============================================================================
//
// Function   : LoadOrdersTable
//
//============================================================================

void LoadOrdersTable(LOADER_TIME_STRUCT *orders_time_start)
{
        int         i;
    long        o_id;
        short          o_d_id;
        short          o_w_id;
    long        o_c_id;
    short       o_carrier_id;
    short       o_ol_cnt;
    short       o_all_local;
        char           o_entry_d[50];

        // bind ORDER data
    bcp_bind(o_dbproc1, (BYTE *) &o_id,          0, -1,          NULL, 0,
0, 1);
    bcp_bind(o_dbproc1, (BYTE *) &o_d_id,        0, -1,          NULL, 0,
0, 2);
    bcp_bind(o_dbproc1, (BYTE *) &o_w_id,        0, -1,          NULL, 0,
0, 3);
    bcp_bind(o_dbproc1, (BYTE *) &o_c_id,        0, -1,          NULL, 0,
0, 4);
    bcp_bind(o_dbproc1, (BYTE *) o_entry_d,      0, 50,          NULL, 0,
SQLCHAR, 5);
    bcp_bind(o_dbproc1, (BYTE *) &o_carrier_id,  0, -1,          NULL, 0,
0, 6);
    bcp_bind(o_dbproc1, (BYTE *) &o_ol_cnt,      0, -1,          NULL, 0,
0, 7);
    bcp_bind(o_dbproc1, (BYTE *) &o_all_local,   0, -1,          NULL, 0,
0, 8);

    for (i = 0; i < ORDERS_PER_DISTRICT; i++)
    {
                o_id         = orders_buf[i].o_id;
                o_d_id       = orders_buf[i].o_d_id;
                o_w_id       = orders_buf[i].o_w_id;
                o_c_id       = orders_buf[i].o_c_id;
                o_carrier_id = orders_buf[i].o_carrier_id;
                o_ol_cnt     = orders_buf[i].o_ol_cnt;
                o_all_local  = orders_buf[i].o_all_local;
                CurrentDate(&o_entry_d);

                // send data to server
                if (!bcp_sendrow(o_dbproc1))
                printf("Error, LoadOrdersTable() failed calling
bcp_sendrow().  Check error file.\n");
                orders_rows_loaded++;
                // CheckForCommit(o_dbproc1, orders_rows_loaded, "ORDERS",
&orders_time_start->time_start);
    }

        bcp_batch(o_dbproc1);

        if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
        {
                bcp_done(o_dbproc1);
                dbclose(o_dbproc1);

                if (aptr->build_index == 1)
                        BuildIndex("idxordcl");
        }

        InterlockedIncrement(&order_threads_completed);

}


//============================================================================
//
// Function   : LoadNewOrderTable
//
//============================================================================

void LoadNewOrderTable(LOADER_TIME_STRUCT *new_order_time_start)
{
    int         i;
    long        o_id;
    short       o_d_id;
    short       o_w_id;

        // Bind NEW-ORDER data
    bcp_bind(o_dbproc2, (BYTE *) &o_id,          0, -1, NULL, 0, 0, 1);
    bcp_bind(o_dbproc2, (BYTE *) &o_d_id,        0, -1, NULL, 0, 0, 2);
    bcp_bind(o_dbproc2, (BYTE *) &o_w_id,        0, -1, NULL, 0, 0, 3);

        for (i = 2100; i < 3000; i++)
        {
                o_id    = orders_buf[i].o_id;
                o_d_id  = orders_buf[i].o_d_id;
                o_w_id  = orders_buf[i].o_w_id;

                if (!bcp_sendrow(o_dbproc2))
                printf("Error, LoadNewOrderTable() failed calling
bcp_sendrow().  Check error file.\n");
                new_order_rows_loaded++;
                // CheckForCommit(o_dbproc2, new_order_rows_loaded,
"NEW_ORDER", &new_order_time_start->time_start);
        }

        bcp_batch(o_dbproc2);
```

```
        if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
        {
                bcp_done(o_dbproc2);
                dbclose(o_dbproc2);

                if (aptr->build_index == 1)
                        BuildIndex("idxnodcl");
        }

        InterlockedIncrement(&order_threads_completed);

}


//============================================================================
//
// Function   : LoadOrderLineTable
//
//============================================================================

void LoadOrderLineTable(LOADER_TIME_STRUCT *order_line_time_start)
{
        int         i,j;
    long        o_id;
        short           o_d_id;
        short           o_w_id;
    long        ol;
        long            ol_i_id;
    short       ol_supply_w_id;
    short       ol_quantity;
    double      ol_amount;
    short       o_all_local;
    char        ol_dist_info[DIST_INFO_LEN+1];
        char            ol_delivery_d[50];

        // bind ORDER-LINE data
    bcp_bind(o_dbproc3, (BYTE *) &o_id,                 0, -1, NULL, 0, 0,
1);
    bcp_bind(o_dbproc3, (BYTE *) &o_d_id,               0, -1, NULL, 0, 0,
2);
    bcp_bind(o_dbproc3, (BYTE *) &o_w_id,               0, -1, NULL, 0, 0,
3);
    bcp_bind(o_dbproc3, (BYTE *) &ol,                   0, -1, NULL, 0, 0,
4);
    bcp_bind(o_dbproc3, (BYTE *) &ol_i_id,              0, -1, NULL, 0, 0,
5);
    bcp_bind(o_dbproc3, (BYTE *) &ol_supply_w_id,       0, -1, NULL, 0, 0,
6);
        bcp_bind(o_dbproc3, (BYTE *) ol_delivery_d,                 0, 50,
NULL, 0, SQLCHAR, 7);
    bcp_bind(o_dbproc3, (BYTE *) &ol_quantity,          0, -1, NULL, 0, 0,
8);
    bcp_bind(o_dbproc3, (BYTE *) &ol_amount,            0, -1, NULL, 0,
SQLFLT8, 9);
    bcp_bind(o_dbproc3, (BYTE *) ol_dist_info,          0, DIST_INFO_LEN,
NULL, 0, 0, 10);

        for (i = 0; i < ORDERS_PER_DISTRICT; i++)
    {
                o_id    = orders_buf[i].o_id;
                o_d_id  = orders_buf[i].o_d_id;
                o_w_id  = orders_buf[i].o_w_id;

                for (j=0; j < orders_buf[i].o_ol_cnt; j++)
                {
                        ol              = orders_buf[i].o_ol[j].ol;
                        ol_i_id         = orders_buf[i].o_ol[j].ol_i_id;
                        ol_supply_w_id =
orders_buf[i].o_ol[j].ol_supply_w_id;
                        ol_quantity     = orders_buf[i].o_ol[j].ol_quantity;
                        ol_amount       = orders_buf[i].o_ol[j].ol_amount;
                        //  Changed to insure ol_delivery_d set properly
(now set in OrdersBufLoad)
                        //  CurrentDate(&ol_delivery_d);

        strcpy(ol_delivery_d,orders_buf[i].o_ol[j].ol_delivery_d);


        strcpy(ol_dist_info,orders_buf[i].o_ol[j].ol_dist_info);

                        if (!bcp_sendrow(o_dbproc3))
                                printf("Error, LoadOrderLineTable() failed
calling bcp_sendrow().  Check error file.\n");
                        order_line_rows_loaded++;
                        // CheckForCommit(o_dbproc3,
order_line_rows_loaded, "ORDER_LINE", &order_line_time_start->time_start);
                }

        }

        bcp_batch(o_dbproc3);

        if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
        {
                bcp_done(o_dbproc3);
                dbclose(o_dbproc3);

                if (aptr->build_index == 1)
                        BuildIndex("idxodlcl");
        }

        InterlockedIncrement(&order_threads_completed);

}


//============================================================================
//
// Function   : GetPermutation
//
//============================================================================

void GetPermutation(int perm[], int n)
{
        int i, r, t;

        for (i=1;i<=n;i++)
                perm[i] = i;
```

```
        for (i=1;i<=n;i++)
        {
                r = RandomNumber(i,n);
                t = perm[i];
                perm[i] = perm[r];
                perm[r] = t;
        }

}


//==============================================================================
//
// Function   : CheckForCommit
//
//==============================================================================

void CheckForCommit(DBPROCESS *dbproc,
                            int rows_loaded,
                                    char *table_name,
                            long *time_start)
{

    long       time_end, time_diff;

    // commit every "batch" rows

    if ( !(rows_loaded % aptr->batch) )
    {
                bcp_batch(dbproc);

                time_end = (TimeNow() / MILLI);
                time_diff = time_end - *time_start;

                printf("-> Loaded %ld rows into %s in %ld sec - Total = %d
(%.2f rps)\n",
                        aptr->batch,
                        table_name,
                                time_diff,
                        rows_loaded,
                                (float) aptr->batch / (time_diff ? time_diff
: 1L));

                *time_start = time_end;
    }

    return;
}


//==============================================================================
//
// Function   : OpenConnections
//
//==============================================================================

void OpenConnections()
{
        RETCODE retcode;

        LOGINREC *login;

    login = dblogin();

    retcode = DBSETLUSER(login, aptr->user);
       if (retcode == FAIL)
       {
                printf("DBSETLUSER failed.\n");
       }
       retcode = DBSETLPWD(login, aptr->password);
       if (retcode == FAIL)
       {
                printf("DBSETLPWD failed.\n");
       }

       retcode = DBSETLPACKET(login, (USHORT) aptr->pack_size);
       if (retcode == FAIL)
       {
                printf("DBSETLPACKET failed.\n");
       }

       printf("DB-Library packet size: %ld\n",aptr->pack_size);

    // turn connection into a BCP connection
    retcode = BCP_SETL(login, TRUE);
       if (retcode == FAIL)
       {
                printf("BCP_SETL failed.\n");
       }

    // open connections to SQL Server */

    if ((i_dbproc1 = dbopen(login, aptr->server)) == NULL)
    {
                printf("Error on login 1 to server %s.\n", aptr->server);
                exit(-1);
    }

    if ((w_dbproc1 = dbopen(login, aptr->server)) == NULL)
    {
                printf("Error on login 2 to server %s.\n", aptr->server);
                exit(-1);
    }

    if ((w_dbproc2 = dbopen(login, aptr->server)) == NULL)
    {
                printf("Error on login 3 to server %s.\n", aptr->server);
                exit(-1);
    }

    if ((c_dbproc1 = dbopen(login, aptr->server)) == NULL)
    {
                printf("Error on login 4 to server %s.\n", aptr->server);
                exit(-1);
    }

    if ((c_dbproc2 = dbopen(login, aptr->server)) == NULL)
    {
                printf("Error on login 5 to server %s.\n", aptr->server);
                exit(-1);
    }
```

```
    if ((o_dbproc1 = dbopen(login, aptr->server)) == NULL)
    {
               printf("Error on login 6 to server %s.\n", aptr->server);
               exit(-1);
    }

    if ((o_dbproc2 = dbopen(login, aptr->server)) == NULL)
    {
               printf("Error on login 7 to server %s.\n", aptr->server);
               exit(-1);
    }

    if ((o_dbproc3 = dbopen(login, aptr->server)) == NULL)
    {
               printf("Error on login 8 to server %s.\n", aptr->server);
               exit(-1);
    }

}



//============================================================================
//
// Function name: SQLErrHandler
//
//============================================================================

int SQLErrHandler(SQLCONN *dbproc,
                           int     severity,
                           int     err,
                           int     oserr,
                           char    *dberrstr,
                           char    *oserrstr)
{

       char msg[256];
       FILE *fp1;
       char timebuf[128];
       char datebuf[128];

       _strtime(timebuf);
       _strdate(datebuf);

       sprintf(msg, "%s %s : DBLibrary (%ld) %s\n", datebuf, timebuf,
err, dberrstr);
    printf("%s",msg);

       fp1 = fopen("logs\tpccldr.err","a");
       if (fp1 == NULL)
       {
              printf("Error in opening errorlog file.\n");
       }
       else
       {
              fprintf(fp1, msg);
              fclose(fp1);
       }

    if (oserr != DBNOERR)
    {
               sprintf(msg, "%s %s : OSError (%ld) %s\n", datebuf,
timebuf, oserr, oserrstr);
               printf("%s",msg);

               fp1 = fopen("logs\\tpccldr.err","a");
               if (fp1 == NULL)
               {
                      printf("Error in opening errorlog file.\n");
               }
               else
               {
                      fprintf(fp1, msg);
                      fclose(fp1);
               }
    }

       if ((dbproc == NULL) || (DBDEAD(dbproc)))
       {
              exit(-1);
       }

       return (INT_CANCEL);

}


//============================================================================
//
// Function name: SQLMsgHandler
//
//============================================================================

int SQLMsgHandler(SQLCONN   *dbproc,
                           DBINT     msgno,
                              int       msgstate,
                              int       severity,
                              char      *msgtext)
{
    char msg[256];
       FILE    *fp1;
       char timebuf[128];
       char datebuf[128];

    if ( (msgno == 5701) || (msgno == 2528) || (msgno == 5703) || (msgno
== 6006) )
       {
              return(INT_CONTINUE);
       }

       if (msgno == 0)
       {
              return(INT_CONTINUE);
       }
       else
       {
              _strtime(timebuf);
              _strdate(datebuf);
```

```
                sprintf(msg, "%s %s : SQLServer (%ld) %s\n", datebuf,
timebuf, msgno, msgtext);


                printf("%s",msg);

                fp1 = fopen("logs\\tpccldr.err","a");
                if (fp1 == NULL)
                {
                        printf("Error in opening errorlog file.\n");
                }
                else
                {
                        fprintf(fp1, msg);
                        fclose(fp1);
                }

                exit(-1);
        }

        return (INT_CANCEL);
}



//============================================================================
//
// Function name: CurrentDate
//
//============================================================================

void CurrentDate(char *datetime)
{

        char timebuf[128];
        char datebuf[128];

        _strtime(timebuf);
        _strdate(datebuf);

        sprintf(datetime, "%s %s", datebuf, timebuf);

}


//============================================================================
//
// Function name: BuildIndex
//
//============================================================================

void BuildIndex(char  *index_script)
{
        char    cmd[256];

        printf("Starting index creation:  %s\n",index_script);

        sprintf(cmd, "isql -S%s -U%s -P%s -e -i%s\\%s.sql >>
logs\\%s.out",
                                aptr->server,
                                aptr->user,
                                aptr->password,
                                aptr->index_script_path,
                                index_script,
                                index_script);

        system(cmd);

        printf("Finished index creation:  %s\n",index_script);
}
```

GETARGS.C

```
// TPC-C Benchmark Kit
//
// Module:  GETARGS.C
// Author:  DamienL


// Includes
#include "tpcc.h"


//============================================================================
//
// Function name: GetArgsLoader
//
//============================================================================

void GetArgsLoader(int argc, char **argv, TPCCLDR_ARGS *pargs)
{
    int        i;
    char       *ptr;

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsLoader()\n", (int)
GetCurrentThreadId());
#endif

    /* init args struct with some useful values */
    pargs->server                   =  SERVER;
    pargs->user                     =  USER;
    pargs->password                 =  PASSWORD;
    pargs->database                 =  DATABASE;
    pargs->batch                    =  BATCH;
    pargs->num_warehouses           =  UNDEF;
    pargs->table                    =  NULL;
       pargs->loader_res_file               =  LOADER_RES_FILE;
       pargs->pack_size                     =  DEFLDPACKSIZE;
       pargs->starting_warehouse    =  DEF_STARTING_WAREHOUSE;
       pargs->build_index                   =  BUILD_INDEX;
       pargs->index_script_path     =  INDEX_SCRIPT_PATH;

    /* check for zero command line args */
    if ( argc == 1 )
               GetArgsLoaderUsage();

    for (i = 1; i < argc; ++i)
    {
```

```
if (argv[i][0] != '-' && argv[i][0] != '/')
{
printf("\nUnrecognized command");
GetArgsLoaderUsage();
exit(1);
}

ptr = argv[i];

switch (ptr[1])
{
case 'h':     /* Fall throught */
case 'H':
               GetArgsLoaderUsage();
               break;

case 'D':
               pargs->database = ptr+2;
               break;

case 'P':
               pargs->password = ptr+2;
               break;

case 'S':
               pargs->server = ptr+2;
               break;

case 'U':
               pargs->user = ptr+2;
               break;

case 'b':
               pargs->batch = atol(ptr+2);
               break;

case 'W':
               pargs->num_warehouses = atol(ptr+2);
               break;

case 's':
               pargs->starting_warehouse = atol(ptr+2);
               break;

case 't':
               pargs->table = ptr+2;
               break;

       case 'f':
               pargs->loader_res_file = ptr+2;
               break;

       case 'p':
               pargs->pack_size = atol(ptr+2);
               break;

       case 'i':
               pargs->build_index = atol(ptr+2);
               break;

               case 'd':
                               pargs->index_script_path = ptr+2;
                               break;

               default:
                       GetArgsLoaderUsage();
                       exit(-1);
                       break;
               }
    }

    /* check for required args */
    if (pargs->num_warehouses == UNDEF )
    {
               printf("Number of Warehouses is required\n");
               exit(-2);
    }

    return;
}


//==============================================================================
//
// Function name: GetArgsLoaderUsage
//
//==============================================================================

void GetArgsLoaderUsage()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsLoaderUsage()\n", (int)
GetCurrentThreadId());
#endif


    printf("TPCCLDR:\n\n");
        printf("Parameter
Default\n");
        printf("-----------------------------------------------------------
----\n");
    printf("-W Number of Warehouses to Load                          Required
\n");
    printf("-S Server                                                %s\n",
SERVER);
    printf("-U Username                                              %s\n",
USER);
    printf("-P Password                                              %s\n",
PASSWORD);
    printf("-D Database                                              %s\n",
DATABASE);
        printf("-b Batch Size
%ld\n", (long) BATCH);
        printf("-p TDS packet size
%ld\n", (long) DEFLDPACKSIZE);
        printf("-f Loader Results Output Filename
%s\n", LOADER_RES_FILE);
        printf("-s Starting Warehouse
%ld\n", (long) DEF_STARTING_WAREHOUSE);
```

```
        printf("-i Build Option (data = 0, data and index = 1)
%ld\n", (long) BUILD_INDEX);
        printf("-d Index Script Path
%s\n", INDEX_SCRIPT_PATH);
        printf("-t Table to Load                                    all
tables \n");
    printf("   [item|warehouse|customer|orders]\n");

        printf("\nNote:  Command line switches are case sensitive.\n");

    exit(0);
}


//==============================================================================
//
// Function name: GetArgsMaster
//
//==============================================================================

void GetArgsMaster(int argc, char **argv, MASTER_DATA *pargs)
{
    int        i;
    char       *ptr;

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsMaster()\n", (int)
GetCurrentThreadId());
#endif

    pargs->server                   = SERVER;
    pargs->database                 = DATABASE;
        pargs->admin_database           = ADMIN_DATABASE;
    pargs->user                     = USER;
    pargs->password                 = PASSWORD;
    pargs->ramp_up                  = RAMP_UP;
    pargs->steady_state             = STEADY_STATE;
    pargs->ramp_down                = RAMP_DOWN;
    pargs->num_users                = NUM_USERS;
    pargs->num_warehouses           = NUM_WAREHOUSES;
    pargs->think_times              = THINK_TIMES;
    pargs->display_data             = DISPLAY_DATA;
        pargs->deadlock_retry           = DEADLOCK_RETRY;
    pargs->tran                     = TRANSACTION;
        pargs->client_mode              = CLIENT_MODE;
        pargs->comment                  = NULL;
        pargs->load_multiplier        = DEF_LOAD_MULTIPLIER;
        pargs->checkpoint_interval      = DEF_CHECKPOINT_INTERVAL;
        pargs->first_checkpoint         = DEF_FIRST_CHECKPOINT;
        pargs->delivery_backoff         = DELIVERY_BACKOFF;
        pargs->num_deliveries           = NUM_DELIVERIES;
        pargs->disable_90th                     = DISABLE_90TH;
        pargs->enable_sqlstat                = ENABLE_SQLSTAT;
        pargs->resfilename                      = RESFILENAME;
        pargs->sqlstat_filename                 = SQLSTAT_FILENAME;
        pargs->sqlstat_period                = SQLSTAT_PERIOD;
        pargs->shutdown_server               = SHUTDOWN_SERVER;
        pargs->auto_run                                 = AUTO_RUN;
        pargs->disable_sqlperf               = DISABLE_SQLPERF;

    /* check for zero command line args */
    if ( argc == 1 )
            GetArgsMasterUsage();

    for (i = 1; i < argc; ++i)
    {
            if (argv[i][0] != '-' && argv[i][0] != '/')
            {
            printf("\nUnrecognized command");
            GetArgsMasterUsage();
            exit(1);
            }

            ptr = argv[i];

            switch (ptr[1])
            {
            case 'h':     /* Fall throught */
                            GetArgsMasterUsage();
                            break;

            case 'S':
                            pargs->server = ptr+2;
                            break;

            case 'D':
                      pargs->database = ptr+2;
                            break;

            case 'A':
                      pargs->admin_database = ptr+2;
                            break;

            case 'U':
                            pargs->user = ptr+2;
                            break;

            case 'P':
                            pargs->password = ptr+2;
                            break;

            case 'u':
                            pargs->ramp_up = atol(ptr+2);
                            break;

            case 's':
                            pargs->steady_state = atol(ptr+2);
                            break;

            case 'd':
                            pargs->ramp_down = atol(ptr+2);
                            break;

            case 'c':
                            pargs->num_users = atol(ptr+2);
                            break;

            case 'w':
                            pargs->num_warehouses = atol(ptr+2);
                            break;

            case 'T':
```

```
                pargs->think_times = atol(ptr+2);
                break;

case 'o':
                pargs->display_data = atol(ptr+2);
                break;

        case 'm':
                pargs->load_multiplier = atof(ptr+2);
                break;

case 'f':
                pargs->first_checkpoint = atol(ptr+2);
                break;

case 'i':
                pargs->checkpoint_interval = atol(ptr+2);
                break;

case 'C':
                pargs->comment = ptr+2;
                break;

case 'B':
                pargs->client_mode = atol(ptr+2);
                break;

case 'n':
                pargs->num_deliveries = atol(ptr+2);
        break;

case 'b':
                pargs->delivery_backoff = atol(ptr+2);
                break;

        case 'r':
                pargs->deadlock_retry = (short) atol(ptr+2);
        break;

        case 't':
                pargs->tran = atol(ptr+2);
                break;

        case 'E':
                pargs->enable_sqlstat = atol(ptr+2);
                break;

case 'e':
                pargs->sqlstat_filename = ptr+2;
                break;

        case 'g':
                pargs->shutdown_server = atol(ptr+2);
                break;

case 'F':
                pargs->resfilename = ptr+2;
                break;

        case 'N':
                pargs->disable_90th = atol(ptr+2);
                break;

                        case 'a':
                                pargs->auto_run = atol(ptr+2);
                                break;

                        case 'q':
                                pargs->disable_sqlperf = atol(ptr+2);
                                break;

                        case 'W':
                                pargs->sqlstat_period = atol(ptr+2);
                                break;

                default:
                                GetArgsMasterUsage();
                                exit(-1);
                                break;
                }
    }

    return;
}


//============================================================================
//
// Function name: GetArgsMasterUsage
//
//============================================================================

void GetArgsMasterUsage()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsMasterUsage()\n", (int)
GetCurrentThreadId());
#endif

    printf("MASTER:\n\n");
        printf("Parameter
Default\n");
        printf("-----------------------------------------------------------
------------\n");
        printf("-S Server
%s\n", SERVER);
    printf("-D Database
%s\n", DATABASE);
        printf("-A Admin Database
%s\n", ADMIN_DATABASE);
    printf("-U Username
%s\n", USER);
    printf("-P Password
%s\n", PASSWORD);
        printf("-u Ramp Up Time (seconds)
%ld\n", (long) RAMP_UP);
        printf("-s Steady State Time (seconds)
%ld\n", (long) STEADY_STATE);
```

```
        printf("-d Ramp Down Time (seconds)
%ld\n", (long) RAMP_DOWN);
    printf("-c Number of Users
%ld\n", (long) NUM_USERS);
        printf("-w Number of Warehouses
%ld\n", (long) NUM_WAREHOUSES);
        printf("-f First Checkpoint (seconds)
%ld\n", (long) DEF_FIRST_CHECKPOINT);
        printf("-i Checkpoint Interval (seconds)
%ld\n", (long) DEF_CHECKPOINT_INTERVAL);
        printf("-B Client mode (TPC-C Scaled = 0, TPC-C Batch = 1)
%ld\n", (long) CLIENT_MODE);
        printf("-n Number of Delivery Threads per Client Driver
%ld\n", (long) NUM_DELIVERIES);
        printf("-b Delivery Queue Backoff Delay (seconds)
%ld\n", (long) DELIVERY_BACKOFF);

        printf("-r Deadlock Retries
%ld\n", (long) DEADLOCK_RETRY);
        printf("-T Use Think Times (no = 0, yes = 1)
%ld\n", (long) THINK_TIMES);
        printf("-m Think Time Load Multiplier
%0.4f\n", DEF_LOAD_MULTIPLIER);
        printf("-o Display Data to Console (no = 0, yes = 1)
%ld\n", (long) DISPLAY_DATA);
        printf("-t Transaction (0, 1, 2, 3, 4, 5)
%ld\n", (long) TRANSACTION);

        printf("-N Disable 90th Per. Calc. (no = 0, yes = 1)
%ld\n",(long) DISABLE_90TH);
        printf("-E Enable Steady State Sqlstats Collection (no = 0, yes =
1)    %ld\n",(long) ENABLE_SQLSTAT);
        printf("-W Sqlstats Collection Period (seconds)
%ld\n",(long) SQLSTAT_PERIOD);
        printf("-e Sqlstats File Name
%s\n", SQLSTAT_FILENAME);
        printf("-g Shutdown SQL Server at End of Test (no = 0, yes = 1)
%ld\n",(long) SHUTDOWN_SERVER);
        printf("-F Result File Name
%s\n", RESFILENAME);
    printf("-a Automated Test Run (no = 0, yes = 1)
%ld\n",(long) AUTO_RUN);
        printf("-C Comment to Include in Result File
None\n");
        printf("\nNote:  Command line switches are case sensitive.\n");

    exit(0);
}


//==============================================================================
//
// Function name: GetArgsClient
//
//==============================================================================

void GetArgsClient(int argc, char **argv, GLOBAL_CLIENT_DATA *pClient)
{
    int        i;
    char       *ptr;
```

```
#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsClient()\n", (int)
GetCurrentThreadId());
#endif

        pClient->num_threads              = NUM_THREADS;
    pClient->server                   = SERVER;
    pClient->database                 = DATABASE;
        pClient->admin_database           = ADMIN_DATABASE;
    pClient->user                     = USER;
    pClient->password                 = PASSWORD;
        pClient->pack_size                = (long) DEFCLPACKSIZE;
        pClient->synch_servername         = SYNCH_SERVERNAME;
        pClient->disable_delivery_resfiles = DISABLE_DELIVERY_RESFILES;
        pClient->enable_qj                              = ENABLE_QJ;

    /* check for 1 or more command line args */
    if ( argc != 1 )
    {
                for (i = 1; i < argc; ++i)
                {
                if (argv[i][0] != '-' && argv[i][0] != '/')
                {
                                printf("\nUnrecognized command");
                                GetArgsClientUsage();
                                exit(1);
                    }

                    ptr = argv[i];

                switch (ptr[1])
                {
                                case 'S':
                                        pClient->server = ptr+2;
                                break;

                                case 'D':
                                        pClient->database = ptr+2;
                                break;

                                case 'A':
                                        pClient->admin_database = ptr+2;
                                break;

                                case 'U':
                                        pClient->user = ptr+2;
                                break;

                                case 'P':
                                        pClient->password = ptr+2;
                                break;

                                case 'c':
                                        pClient->num_threads = atol(ptr+2);
                                break;

                                case 'p':
                                        pClient->pack_size = atol(ptr+2);
                                        break;

                                case 'd':
```

```
                                        pClient->disable_delivery_resfiles =
atol(ptr+2);
                                        break;

                                case 's':
                                        pClient->synch_servername = ptr+2;
                                        break;

                                case 'q':
                                        pClient->enable_qj = atol(ptr+2);
                                        break;

                                default:
                                        GetArgsClientUsage();
                                        exit(-1);
                                        break;
                        }
                        }
                }

        return;
}


//==========================================================================
//
// Function name: GetArgsClientUsage
//
//==========================================================================

void GetArgsClientUsage()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsClientUsage()\n", (int)
GetCurrentThreadId());
#endif

    printf("CLIENT:\n\n");
        printf("Parameter
Default\n");
        printf("-----------------------------------------------------------
-\n");
        printf("-S Server                                               %s\n",
SERVER);
    printf("-D Database                                             %s\n",
DATABASE);
        printf("-A Admin Database                                       %s\n",
ADMIN_DATABASE);
    printf("-U Username                                             %s\n",
USER);
    printf("-P Password                                             %s\n",
PASSWORD);
        printf("-c Number of User Connections
%ld\n",(long) NUM_THREADS);
        printf("-p TDS Packet Size
%ld\n",(long) DEFCLPACKSIZE);
        printf("-d Disable Delivery Result Files (no = 0, yes = 1)
%ld\n",(long) DISABLE_DELIVERY_RESFILES);
        printf("-s Master Driver Servername                             %s\n",
SYNCH_SERVERNAME);

    printf("\nNote:  Command line switches are case sensitive.\n");

    exit(0);
}


//==========================================================================
//
// Function name: GetArgsDelivery
//
//==========================================================================
void GetArgsDelivery(int argc, char **argv, DELIVERY_ARGS *pDelivery)
{
    int        i;
    char       *ptr;

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsDelivery()\n", (int)
GetCurrentThreadId());
#endif

        pDelivery->pipe_num = 0;

    /* check for 1 or more command line args */
    if ( argc != 1 )
        {
                for (i = 1; i < argc; ++i)
                {
                if (argv[i][0] != '-' && argv[i][0] != '/')
                {
                                printf("\nUnrecognized command");
                                GetArgsClientUsage();
                                exit(1);
                }

                ptr = argv[i];

                    switch (ptr[1])
                    {
                                case 'p':
                                        pDelivery->pipe_num = (long)
atol(ptr+2);
                                        break;

                                default:
                                        printf("ERROR:  No pipe number
specified.");
                                        exit(-1);
                                        break;
                }
                }
    }

    return;
}
```

```
//==============================================================================
//
// Function name: GetArgsSQLStat
//
//==============================================================================

void GetArgsSQLStat(int argc, char **argv, SQLSTAT_ARGS *pargs)
{
    int       i;
    char      *ptr;

    /* init args struct with some useful values */
    pargs->server                   =  SERVER;
    pargs->user                   =  USER;
    pargs->password                 =  PASSWORD;
    pargs->admin_database       =  ADMIN_DATABASE;
       pargs->sqlstat_filename     =  SQLSTAT_FILENAME;
       pargs->run_id                       =  UNDEF;

    /* check for zero command line args */
    if ( argc == 1 )
               GetArgsSQLStatUsage();

    for (i = 1; i < argc; ++i)
    {
               if (argv[i][0] != '-' && argv[i][0] != '/')
               {
               printf("\nUnrecognized command");
               GetArgsSQLStatUsage();
               exit(1);
               }

               ptr = argv[i];

               switch (ptr[1])
               {

               case 'S':
                              pargs->server = ptr+2;
                              break;

               case 'U':
                              pargs->user = ptr+2;
                              break;

               case 'P':
                              pargs->password = ptr+2;
                              break;

               case 'A':
                              pargs->admin_database = ptr+2;
                              break;

                      case 'i':
                              pargs->run_id = atol(ptr+2);
                              break;

                      case 'f':
                              pargs->sqlstat_filename = ptr+2;
                              break;

               default:
                              GetArgsSQLStatUsage();
                              exit(-1);
                              break;
               }
    }

    /* check for required args */
    if (pargs->run_id == UNDEF )
    {
               printf("Error, Run ID is required.\n");
               exit(-2);
    }

    return;
}


//==============================================================================
//
// Function name: GetArgsSQLStatUsage
//
//==============================================================================

void GetArgsSQLStatUsage()
{

    printf("SQLSTAT:\n\n");
       printf("Parameter
Default\n");
       printf("-------------------------------------------------------------
----\n");
    printf("-S Server                                                  %s\n",
SERVER);
    printf("-U Username                                                %s\n",
USER);
    printf("-P Password                                                %s\n",
PASSWORD);
    printf("-A Admin Database                                          %s\n",
ADMIN_DATABASE);
       printf("-i Run ID
(required)\n");
       printf("-f Statistics Result file
%s\n", SQLSTAT_FILENAME);

       printf("\nNote:  Command line switches are case sensitive.\n");

    exit(0);
}
```

RANDOM.C

```
/*      FILE:          RANDOM.C
 *                             Microsoft TPC-C Kit Ver. 3.00.000
 *                             Audited 08/23/96, By Francois Raab
 *
 *                             Copyright Microsoft, 1996
 *
```

```
 *      PURPOSE:        Random number generation functions for Microsoft
TPC-C Benchmark Kit
 *      Author:         Damien Lindauer
 *                              damienl@Microsoft.com
 */

// Includes
#include "tpcc.h"
#include "math.h"

// Defines
#define A             16807
#define M        2147483647
#define Q           127773      /* M div A */
#define R             2836      /* M mod A */
#define Thread         __declspec(thread)

// Globals
long    Thread Seed = 0;        /* thread local seed */

/*****************************************************************************
*
*
* random -
*
*       Implements a GOOD pseudo random number generator.  This generator
*
*       will/should? run the complete period before repeating.
*
*
*
* Copied from:
*
*       Random Numbers Generators: Good Ones Are Hard to Find.
*
*       Communications of the ACM - October 1988 Volume 31 Number 10
*
*
*
* Machine Dependencies:
*
*       long must be 2 ^ 31 - 1 or greater.
*
*
*
*****************************************************************************
****/

/*****************************************************************************
*****
* seed - load the Seed value used in irand and drand.  Should be used
before  *
*       first call to irand or drand.
*
*****************************************************************************
****/

void seed(long val)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering seed()...\n", (int) GetCurrentThreadId());
        printf("Old Seed %ld New Seed %ld\n",Seed, val);
#endif

    if ( val < 0 )
           val = abs(val);

    Seed = val;

}


/*****************************************************************************
****
*
*
* irand - returns a 32 bit integer pseudo random number with a period of
*
*       1 to 2 ^ 32 - 1.
*
*
*
* parameters:
*
*       none.
*
*
*
* returns:
*
*       32 bit integer - defined as long ( see above ).
*
*
*
* side effects:
*
*       seed get recomputed.
*
*****************************************************************************
***/

long irand()
{
    register long    s;      /* copy of seed */
    register long    test;   /* test flag */
    register long    hi;     /* tmp value for speed */
    register long    lo;     /* tmp value for speed */

#ifdef DEBUG
    printf("[%ld]DBG: Entering irand()...\n", (int) GetCurrentThreadId());
#endif

    s = Seed;
    hi = s / Q;
    lo = s % Q;

    test = A * lo - R * hi;
    if ( test > 0 )
```

```
                Seed = test;
    else
                Seed = test + M;

    return( Seed );
}


/*************************************************************************
****
*
*
* drand - returns a double pseudo random number between 0.0 and 1.0.
*
*       See irand.
*
*************************************************************************
***/
double drand()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering drand()...\n", (int) GetCurrentThreadId());
#endif

    return( (double)irand() / 2147483647.0);
}


//=========================================================================
// Function   : RandomNumber
//
// Description:
//=========================================================================
long RandomNumber(long lower, long upper)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering RandomNumber()...\n", (int)
GetCurrentThreadId());
#endif

        if ( upper == lower )  /* pgd 08-13-96 perf enhancement */
                return lower;

        upper++;

    if ( upper <= lower )
                rand_num = upper;
    else
                rand_num = lower + irand() % (upper - lower); /* pgd 08-13-
96 perf enhancement */


#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber between %ld & %ld ==> %ld\n",
                            (int) GetCurrentThreadId(), lower, upper,
rand_num);
#endif

    return rand_num;
}


#if 0

//Orginal code pgd 08/13/96

long RandomNumber(long lower,
                                        long upper)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering RandomNumber()...\n", (int)
GetCurrentThreadId());
#endif

        upper++;

    if ((upper <= lower))
                rand_num = upper;
    else
                rand_num = lower + irand() % ((upper > lower) ? upper -
lower : upper);


#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber between %ld & %ld ==> %ld\n",
                                (int) GetCurrentThreadId(), lower, upper,
rand_num);
#endif


    return rand_num;
}
#endif

//=========================================================================
// Function   : NURand
//
// Description:
//=========================================================================
long NURand(int iConst,
            long x,
            long y,
            long C)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering NURand()...\n", (int)
GetCurrentThreadId());
#endif

    rand_num = (((RandomNumber(0,iConst) | RandomNumber(x,y)) + C) % (y-
x+1))+x;

#ifdef DEBUG
```

```
    printf("[%ld]DBG: NURand: num = %d\n", (int) GetCurrentThreadId(),
rand_num);
#endif

    return rand_num;
}
```

UTIL.C

```
// TPC-C Benchmark Kit
//
// Module:  UTIL.C
// Author:  DamienL


// Includes
#include "tpcc.h"


//============================================================================
//
// Function name: UtilSleep
//
//============================================================================

void UtilSleep(long  delay)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilSleep()\n", (int)
GetCurrentThreadId());
#endif

#ifdef DEBUG
    printf("[%ld]DBG: Sleeping for %ld seconds...\n", (int)
GetCurrentThreadId(), delay);
#endif

    Sleep(delay * 1000);

}


//============================================================================
//
// Function name: UtilSleep
//
//============================================================================

void UtilSleepMs(long  delay)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilSleepMs()\n", (int)
GetCurrentThreadId());
#endif

#ifdef DEBUG
    printf("[%ld]DBG: Sleeping for %ld milliseconds...\n", (int)
GetCurrentThreadId(), delay);
#endif

    Sleep(delay);

}

//============================================================================
//
// Function name: UtilPrintNewOrder
//
//============================================================================
void UtilPrintNewOrder(NEW_ORDER_DATA *pNewOrder)
{
    int i;

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintNewOrder()\n", (int)
GetCurrentThreadId());
#endif

        EnterCriticalSection(&ConsoleCritSec);

        printf("\n[%04ld]\tNewOrder Transaction\n\n", (int)
GetCurrentThreadId());

    printf("Warehouse: %ld\n"
                "District: %ld\n"
                "Date: %02ld/%02ld/%04ld %02ld:%02ld:%02ld\n\n"
                "Customer Number:  %ld\n"
                "Customer Name: %s\n"
                "Customer Credit: %s\n"
                "Cusotmer Discount: %02.2f%%\n\n"
                "Order Number: %ld\n"
                "Warehouse Tax: %02.2f%%\n"
                "District Tax: %02.2f%%\n\n"
                "Number of Order Lines: %ld\n\n",
                    (int)    pNewOrder->w_id,
                    (int)    pNewOrder->d_id,
                    (char *) pNewOrder->o_entry_d.month,
                    (char *) pNewOrder->o_entry_d.day,
                    (char *) pNewOrder->o_entry_d.year,
                    (char *) pNewOrder->o_entry_d.hour,
                    (char *) pNewOrder->o_entry_d.minute,
                    (char *) pNewOrder->o_entry_d.second,
                    (int)    pNewOrder->c_id,
                    (char *) pNewOrder->c_last,
                    (char *) pNewOrder->c_credit,
                    (float)  pNewOrder->c_discount,
                    (int)    pNewOrder->o_id,
                    (float)  pNewOrder->w_tax,
                    (float)  pNewOrder->d_tax,
                    (int)    pNewOrder->o_ol_cnt);

        printf("Supp_W Item_Id Item Name                  Qty Stock B/G
Price    Amount   \n");
        printf("------ ------- ------------------------ --- ----- --- --
------ ---------\n");
```

```
    for (i=0;i < pNewOrder->o_ol_cnt;i++)
    {
                printf("%04ld   %06ld  %24s   %02ld   %03ld   %1s  %8.2f
%9.2f\n",
                (int)    pNewOrder->Ol[i].ol_supply_w_id,
                (int)    pNewOrder->Ol[i].ol_i_id,
                        (char *) pNewOrder->Ol[i].ol_i_name,
                (int)    pNewOrder->Ol[i].ol_quantity,
                (int)    pNewOrder->Ol[i].ol_stock,
                (char *) pNewOrder->Ol[i].ol_brand_generic,
                (float)  pNewOrder->Ol[i].ol_i_price,
                (float)  pNewOrder->Ol[i].ol_amount);
    }

        printf("\nTotal: $%05.2f\n\n",
                        (float)  pNewOrder->total_amount);

        printf("Execution Status: %s\n\n",
                        (char *) pNewOrder->execution_status);

        LeaveCriticalSection(&ConsoleCritSec);

}


//============================================================================
//
// Function name: UtilPrintPayment
//
//============================================================================

void UtilPrintPayment(PAYMENT_DATA *pPayment)
{

        char    tmp_data[201];
    char    data_line_1[51];
        char    data_line_2[51];
        char    data_line_3[51];
        char    data_line_4[51];

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintPayment()\n", (int)
GetCurrentThreadId());
#endif

    EnterCriticalSection(&ConsoleCritSec);

        printf("\n[%04ld]\tPayment Transaction\n\n", (int)
GetCurrentThreadId());

    printf("Date: %02ld/%02ld/%04ld %02ld:%02ld:%02ld\n\n",
                (int)    pPayment->h_date.month,
                        (int)    pPayment->h_date.day,
                (int)    pPayment->h_date.year,
                        (int)    pPayment->h_date.hour,
                        (int)    pPayment->h_date.minute,
                        (int)    pPayment->h_date.second);

        printf("Warehouse: %ld\n"
                   "District: %ld\n\n",
                        (int)    pPayment->w_id,
                        (int)    pPayment->d_id);

        printf("Warehouse Address Street 1: %s\n"
                  "Warehouse Address Street 2: %s\n",
                        (char *) pPayment->w_street_1,
                        (char *) pPayment->w_street_2);

        printf("Warehouse Address City: %s\n"
               "Warehouse Address State: %s\n"
               "Warehouse Address Zip: %s\n\n\n",
                        (char *) pPayment->w_city,
                        (char *) pPayment->w_state,
                        (char *) pPayment->w_zip);

        printf("District Address Street 1: %s\n"
                  "District Address Street 2: %s\n",
                        (char *) pPayment->d_street_1,
                    (char *) pPayment->d_street_2);

        printf("District Address City: %s\n"
               "District Address State: %s\n"
               "District Address Zip: %s\n\n\n",
                    (char *) pPayment->d_city,
                        (char *) pPayment->d_state,
                        (char *) pPayment->d_zip);

        printf("Customer Number: %ld\n"
                  "Customer Warehouse: %ld\n"
                  "Customer District: %ld\n",
                        (int)    pPayment->c_id,
                        (int)    pPayment->c_w_id,
                        (int)    pPayment->c_d_id);

        printf("Customer Name:   %s %s %s\n"
                  "Customer Since:  %02ld-%02ld-%04ld\n",
                        (char *) pPayment->c_first,
                        (char *) pPayment->c_middle,
                        (char *) pPayment->c_last,
                        (int)    pPayment->c_since.month,
                        (int)    pPayment->c_since.day,
                        (int)    pPayment->c_since.year);

        printf("Customer Address Street 1: %s\n"
               "Customer Address Street 2: %s\n"
               "Customer Address City: %s\n"
               "Customer Address State: %s\n"
               "Customer Address Zip: %s\n"
                  "Customer Phone Number: %s\n\n\n"
               "Customer Credit: %s\n"
               "Customer Discount: %02.2f%%\n",
                        (char *) pPayment->c_street_1,
                        (char *) pPayment->c_street_2,
                        (char *) pPayment->c_city,
                        (char *) pPayment->c_state,
                        (char *) pPayment->c_zip,
                        (char *) pPayment->c_phone,
                        (char *) pPayment->c_credit,
                        (double) pPayment->c_discount);

        printf("Amount Paid: $%04.2f\n"
```

```
                "New Customer Balance: $%10.2f\n",
                        (float)  pPayment->h_amount,
                        (double) pPayment->c_balance);

    printf("Credit Limit: $%10.2f\n\n",
                        (double) pPayment->c_credit_lim);

    if (strcmp(pPayment->c_data," ") != 0)
        {
                strcpy(tmp_data, pPayment->c_data);
                strncpy(data_line_1, tmp_data, 50);       data_line_1[50] =
'\0';
                strncpy(data_line_2, &tmp_data[50], 50);  data_line_2[50] =
'\0';
                strncpy(data_line_3, &tmp_data[100], 50); data_line_3[50] =
'\0';
                strncpy(data_line_4, &tmp_data[150], 50); data_line_4[50] =
'\0';

        }
        else
        {
                strcpy(data_line_1, " "); strcpy(data_line_2, " ");
                strcpy(data_line_3, " "); strcpy(data_line_4, " ");
        }

        printf("                ----------------------------------------
--------\n");
    printf("Customer Data: |%50s|\n", data_line_1);
        printf("               |%50s|\n", data_line_2);
        printf("               |%50s|\n", data_line_3);
        printf("               |%50s|\n", data_line_4);
        printf("                ----------------------------------------
--------\n\n");

        printf("Execution Status: %s\n\n",
                        (char *) pPayment->execution_status);

        LeaveCriticalSection(&ConsoleCritSec);

}


//============================================================================
//
// Function name: UtilPrintOrderStatus
//
//============================================================================

void UtilPrintOrderStatus(ORDER_STATUS_DATA *pOrderStatus)
{
    int i;

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintOrderStatus()\n", (int)
GetCurrentThreadId());
#endif

        EnterCriticalSection(&ConsoleCritSec);

        printf("\n[%04ld]\tOrder-Status Transaction\n\n", (int)
GetCurrentThreadId());

        printf("Warehouse: %ld\n"
                "District: %ld\n\n",
                        (int) pOrderStatus->w_id,
                        (int) pOrderStatus->d_id);

        printf("Customer Number: %ld\n"
                "Customer Name: %s %s %s\n\n",
                        (int) pOrderStatus->c_id,
                        (char *) pOrderStatus->c_first,
                        (char *) pOrderStatus->c_middle,
                        (char *) pOrderStatus->c_last);

        printf("Customer Balance: $%5.2f\n\n",
                        (double) pOrderStatus->c_balance);

        printf("Order Number: %ld\n"
                "Entry Date: %02ld/%02ld/%04ld %02ld:%02ld:%02ld\n"
                "Carrier Number: %ld\n\n"
                "Number of order lines: %ld\n\n",
                        (int) pOrderStatus->o_id,
                        (int) pOrderStatus->o_entry_d.month,
                        (int) pOrderStatus->o_entry_d.day,
                        (int) pOrderStatus->o_entry_d.year,
                        (int) pOrderStatus->o_entry_d.hour,
                        (int) pOrderStatus->o_entry_d.minute,
                        (int) pOrderStatus->o_entry_d.second,
                        (int) pOrderStatus->o_carrier_id,
                        (int) pOrderStatus->o_ol_cnt);

    printf ("Supply-W    Item-Id    Delivery-Date    Qty   Amount    \n");
        printf ("--------    -------    -------------    ---   ----------
\n");

    for (i=0;i < pOrderStatus->o_ol_cnt; i++)
    {
                printf("%04ld        %06ld     %02ld/%02ld/%04ld
%02ld     %9.2f\n",
                                (int) pOrderStatus-
>OlOrderStatusData[i].ol_supply_w_id,
                                (int) pOrderStatus-
>OlOrderStatusData[i].ol_i_id,
                                (int) pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d.month,
                                (int) pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d.day,
                                (int) pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d.year,
                                (int) pOrderStatus-
>OlOrderStatusData[i].ol_quantity,
                                (double) pOrderStatus-
>OlOrderStatusData[i].ol_amount);
    }

        if (pOrderStatus->o_ol_cnt == 0)
                printf("\nNo Order-Status items.\n\n");

        printf("\nExecution Status: %s\n\n",
```

```
                        (char *) pOrderStatus->execution_status);

        LeaveCriticalSection(&ConsoleCritSec);

}


//================================================================================
//
// Function name: UtilPrintDelivery
//
//================================================================================

void UtilPrintDelivery(DELIVERY_DATA *pQueuedDelivery)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintDelivery()\n", (int)
GetCurrentThreadId());
#endif

        EnterCriticalSection(&ConsoleCritSec);

        printf("\n[%04ld]\tDelivery Transaction\n\n", (int)
GetCurrentThreadId());

    printf("Warehouse: %ld\n", (int) pQueuedDelivery->w_id);

        printf("Carrier Number: %ld\n\n", (int) pQueuedDelivery-
>o_carrier_id);

    printf("Execution Status: %s\n\n", (char *) pQueuedDelivery-
>execution_status);

        LeaveCriticalSection(&ConsoleCritSec);

}


//================================================================================
//
// Function name: UtilPrintStockLevel
//
//================================================================================

void UtilPrintStockLevel(STOCK_LEVEL_DATA *pStockLevel)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilPrintStockLevel()\n", (int)
GetCurrentThreadId());
#endif

        EnterCriticalSection(&ConsoleCritSec);

        printf("\n[%04ld]\tStock-Level Transaction\n\n", (int)
GetCurrentThreadId());

        printf("Warehouse: %ld\nDistrict: %ld\n",
                        (int) pStockLevel->w_id,
                        (int) pStockLevel->d_id);

        printf("Stock Level Threshold: %ld\n\n", (int) pStockLevel-
>thresh_hold);

        printf("Low Stock Count: %ld\n\n", (int) pStockLevel->low_stock);

        printf("Execution Status: %s\n\n", (char *) pStockLevel-
>execution_status);

        LeaveCriticalSection(&ConsoleCritSec);
}

//================================================================================
//
// Function name: UtilError
//
//================================================================================
void UtilError(long threadid, char * header, char *msg)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilError()\n", (int)
GetCurrentThreadId());
#endif

    printf("[%ld] %s: %s\n", (int) threadid, header, msg);
}


//================================================================================
//
// Function name: UtilFatalError
//
//================================================================================

void UtilFatalError(long threadid, char * header, char *msg)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilFatalError()\n", (int)
GetCurrentThreadId());
#endif

    printf("[Thread: %ld]... %s: %s\n", (int) threadid, header, msg);
    exit(-1);
}


//================================================================================
//
// Function name: UtilStrCpy
//
//================================================================================
void UtilStrCpy(char * pDest, char * pSrc, int n)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilStrCpy()\n", (int)
GetCurrentThreadId());
#endif
```

```
    strncpy(pDest, pSrc, n);
    pDest[n] = '\0';
}


#ifdef USE_CONMON
//============================================================================
//
// Function name: WriteConsoleString
//
//============================================================================
void WriteConsoleString(HANDLE hConMon, char *str, short x, short y, short
color, BOOL pad)
{
       COORD   dwWriteCoord = {0, 0};
       DWORD   cCharsWritten;
       LPVOID  dummy;
       int     len, i;

#ifdef DEBUG
    printf("[%ld]DBG: Entering WriteConsoleString()\n", (int)
GetCurrentThreadId());
#endif

       dwWriteCoord.X = x;
       dwWriteCoord.Y = y;

       if (pad)
       {
              len = strlen(str);
              if (len < CON_LINE_SIZE)
              {
                     for(i=1;i<CON_LINE_SIZE-len;i++)
                     {
                            strcat(str," ");
                     }
              }
       }

       EnterCriticalSection(&ConsoleCritSec);

       switch (color)
    {
              case YELLOW:
                     SetConsoleTextAttribute(hConMon,
                            FOREGROUND_INTENSITY | FOREGROUND_GREEN |
FOREGROUND_RED | BACKGROUND_BLUE);
              break;

              case RED:
                     SetConsoleTextAttribute(hConMon,
                            FOREGROUND_INTENSITY | FOREGROUND_RED |
BACKGROUND_BLUE);
              break;

              case GREEN:
                     SetConsoleTextAttribute(hConMon,
                            FOREGROUND_INTENSITY | FOREGROUND_GREEN |
BACKGROUND_BLUE);
                     break;
    }

       SetConsoleCursorPosition(hConMon, dwWriteCoord);
       WriteConsole(hConMon, str, strlen(str), &cCharsWritten, dummy);

       LeaveCriticalSection(&ConsoleCritSec);

}
#endif


//============================================================================
//
// Function name: AddDeliveryQueueNode
//
//============================================================================

BOOL AddDeliveryQueueNode(DELIVERY_PTR node_to_add)
{

       DELIVERY_PTR   local_node;
#ifdef DEBUG
       DELIVERY_PTR   ptrtmp;
       short           i;
#endif

       EnterCriticalSection(&QueuedDeliveryCritSec);

       if ((local_node = malloc(sizeof(struct delivery_node)) ) == NULL)
       {
              printf("ERROR:  problem allocating memory for delivery
queue.\n");
              exit(-1);
       }
       else
       {
              memcpy(local_node, node_to_add, sizeof (struct
delivery_node));

              if (queued_delivery_cnt == 0)
              {
                     delivery_head = local_node;
                     delivery_head->next_delivery = NULL;
                     delivery_tail = delivery_head;
              }
              else
              {
                     local_node->next_delivery = NULL;
                     delivery_tail->next_delivery = local_node;
                     delivery_tail = local_node;
              }
       }

       queued_delivery_cnt++;

#ifdef DEBUG
       i=0;
       printf("Add to delivery list: %ld\n",queued_delivery_cnt);
       ptrtmp=delivery_head;
       while (ptrtmp != NULL)
```

```
        {
                i++;
                printf("%ld - w_id %ld - o_carrier_id %ld - queue_time
%d/%d/%d %d:%d:%d:%d\n",
                                i, ptrtmp->w_id, ptrtmp->o_carrier_id,
                                ptrtmp->queue_time.wMonth,
                                ptrtmp->queue_time.wDay,
                                ptrtmp->queue_time.wYear,
                                ptrtmp->queue_time.wHour,
                                ptrtmp->queue_time.wMinute,
                                ptrtmp->queue_time.wSecond,
                                ptrtmp->queue_time.wMilliseconds);

                ptrtmp=ptrtmp->next_delivery;
        }
#endif

        LeaveCriticalSection(&QueuedDeliveryCritSec);

    return TRUE;

}


//============================================================================
//
// Function name: GetDeliveryQueueNode
//
//============================================================================

BOOL GetDeliveryQueueNode(DELIVERY_PTR node_to_get)
{

        DELIVERY_PTR          local_node;
        BOOL                        rc;
#ifdef DEBUG
        DELIVERY_PTR          ptrtmp;
        short                       i;
#endif

        EnterCriticalSection(&QueuedDeliveryCritSec);

        if (queued_delivery_cnt == 0)
        {

#ifdef DEBUG
                printf("No delivery nodes found.\n");
#endif
                rc = FALSE;
        }
        else
        {

                memcpy(node_to_get, delivery_head, sizeof(struct
delivery_node));

                if (queued_delivery_cnt == 1)
                {
                        free(delivery_head);
                        delivery_head = NULL;
                        queued_delivery_cnt = 0;
                }
                else
                {
                        local_node = delivery_head;
                        delivery_head = delivery_head->next_delivery;
                        free(local_node);
                        queued_delivery_cnt--;
                }

#ifdef DEBUG
                i=0;
                printf("Get from delivery list:
%ld\n",queued_delivery_cnt);
                ptrtmp=delivery_head;
                while (ptrtmp != NULL)
                {
                        i++;
                        printf("%ld - w_id %ld - o_carrier_id %ld -
queue_time %d/%d/%d %d:%d:%d:%d\n",
                                        i, ptrtmp->w_id, ptrtmp-
>o_carrier_id,
                                        ptrtmp->queue_time.wMonth,
                                        ptrtmp->queue_time.wDay,
                                        ptrtmp->queue_time.wYear,
                                        ptrtmp->queue_time.wHour,
                                        ptrtmp->queue_time.wMinute,
                                        ptrtmp->queue_time.wSecond,
                                        ptrtmp->queue_time.wMilliseconds);

                        ptrtmp=ptrtmp->next_delivery;
                }
#endif

                rc = TRUE;

        }

        LeaveCriticalSection(&QueuedDeliveryCritSec);

    return rc;

}


//============================================================================
//
// Function name: WriteDeliveryString
//
//============================================================================

void WriteDeliveryString(char   buf[255])
{

        DWORD   bytesWritten;
        DWORD   retCode;

#ifdef DEBUG
    printf("[%ld]DBG: Entering UtilDeliveryMsg()\n", (int)
GetCurrentThreadId());
#endif
```

```
        EnterCriticalSection(&WriteDeliveryCritSec);

    retCode = WriteFile (hDeliveryMonPipe, buf, PLEASE_WRITE,
                         &bytesWritten, NULL);

        LeaveCriticalSection(&WriteDeliveryCritSec);

}
```

Appendix C - Tunable Parameters

Microsoft Windows NT 4.0 Configuration Parameters

There were no Windows NT Registry parameters that were changed from their default settings. The following services were disabled in the Windows NT Control Panel/ Services on the Server:

- License Logging Service
- NT LM Security Support Provider
- Plug and Play
- Spooler

Microsoft SQL Server Startup Parameters

```
c:\ mssql\binn\sqlservr -c -x -t1081 -T812 -T3502 -T1140 -Cd1450000 -
Cp4500
```

Where:

- -c Start SQL Server independently of the Service Control Manager
- -x Disables the keeping of CPU time and cache hit ratio statistics
- -t1081 Allows the index pages a “second” trip through the cache
- -T812 Disables checkpoint buffer sorting
- -T3502 Writes a message to the SQL Server Errorlog showing the beginning and ending time of each checkpoint
- -T1140 Optimizes free space allocation
- -Cd1450000 Defines the number of 2KB database cache buffers
- -Cp4500 Defines the number of buffers for the procedure cache

SQL Server Stack Size

The default stack size for Microsoft SQL Server 6.5.SP4 (6.50.258) was changed using the EDITBIN utility. The EDITBIN utility ships with Microsoft Visual C++ V4.0. The command used to change the stack size is:

```
editbin /S: 65536 sqlservr.exe
```

This command is fully documented as an article in the Microsoft Knowledge Base on the Microsoft Web Site at www.microsoft.com/support.

DBCC GAMINIT

Prior to the execution of the benchmark, the following script was run to proactively populate the Global Allocation Map (GAM) rather than allowing it to be populated on an as needed basis.

```
Use tpcc
go
dbcc gaminit
go
```

This command is fully documented as an article in the Microsoft Knowledge Base on the Microsoft Web Site at www.microsoft.com/support.

‘Cache’ Column of Sysobjects Table

Prior to the execution of the benchmark, the following script was run and SQL Server was restarted to improve cache performance of tables which are accessed non-uniformly. Use of this feature is fully documented as an article in the Microsoft Knowledge Base on the Microsoft Web Site at www.microsoft.com/support.

```
Use tpcc
go
update sysobjects set cache=2 from sysobjects where name='stock'
go
update sysobjects set cache=5 from sysobjects where name='customer'
go
```

BOOT.INI

The /3gb switch was added to the boot.ini file to cause NT Server to allow 3GB of user and 1GB of kernel virtual address space, rather than the usual 2GB of virtual address space for each.

Microsoft SQL Server Startup Parameters

```
Key Name:          SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\Parameters
Class Name:        <NO CLASS>
Last Write Time:   6/25/97 - 9:42 PM
Value 0
  Name:            SQLArg0
  Type:            REG_SZ
  Data:            -dC:\MSSQL\DATA\MASTER.DAT

Value 1
  Name:            SQLArg1
  Type:            REG_SZ
  Data:            -eC:\MSSQL\LOG\ERRORLOG
```

Microsoft SQL Server Configuration Parameters

```
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14>
/*    File:    VERSION.SQL
*/
/*             Microsoft TPC-C Kit Ver. 3.00.000
*/
/*             Audited 08/23/96, By Francois Raab
*/
/*
*/
/*             Copyright Microsoft, 1996
*/
/*
*/
/*    Author:  Damien Lindauer
*/
/*             damienl@Microsoft.com
*/


print " "
select convert(char(30), getdate(),9)
print " "

 ------------------------------
 Oct  7 1997  9:29:02:043AM

(1 row affected)

1> 2> 3>
select @@version


 ------------------------------------------------------------------------------
-----
       ------------------------------------------
 Microsoft SQL Server  6.50 - 6.50.258 (Intel X86)
       Jun 16 1997 11:46:49
       Cop
       yright (c) 1988-1997 Microsoft Corporation


(1 row affected)
1> 2>
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11> 12> 13> 14>
/*    File:    CONFIG.SQL
*/
/*             Microsoft TPC-C Kit Ver. 3.00.000
*/
/*             Audited 08/23/96, By Francois Raab
*/
/*
*/
/*             Copyright Microsoft, 1996
*/
/*
*/
/*    Author:  Damien Lindauer
*/
/*             damienl@Microsoft.com
*/


print " "
select convert(char(30), getdate(),9)
print " "

 ------------------------------
 Oct  7 1997  9:29:02:826AM

(1 row affected)

1> 2> 3> Configuration option changed. Run the RECONFIGURE command to
install.

sp_configure "show advanced",1
1> 2> reconfigure with override
1> 2> sp_configure
```

```
name                                minimum     maximum     config_value
      run_value
----------------------------------- ----------- ----------- ------------
      -----------
affinity mask                             0  2147483647           63
             63
allow updates                             0           1            1
              1
backup buffer size                        1          32            2
              2
backup threads                            0          32            4
              4
cursor threshold                         -1  2147483647           -1
             -1
database size                             2       10000            2
              2
default language                          0        9999            0
              0
default sortorder id                      0         255           50
             50
fill factor                               0         100            0
              0
free buffers                             20      524288         3000
           3000
hash buckets                           4999     1000003       749011
         749011
language in cache                         3         100            3
              3
LE threshold maximum                      2      500000          301
            301
LE threshold minimum                      2      500000           20
             20
LE threshold percent                      1         100            0
              0
locks                                  5000  2147483647         6000
           6000
LogLRU buffers                            0  2147483647         2100
           2100
logwrite sleep (ms)                      -1         500           -1
             -1
max async IO                              1        1024           20
             20
max lazywrite IO                          1        1024          100
            100
max text repl size                        0  2147483647        65536
          65536
max worker threads                       10        1024          230
            230
media retention                           0         365            0
              0
memory                                 2800     1048576       950000
         950000
nested triggers                           0           1            1
              1
network packet size                     512       32767         4096
           4096
open databases                            5       32767           10
             10
open objects                            100  2147483647          450
            450
priority boost                            0           1            0
              0
procedure cache                           1          99            1
              1
Protection cache size                     1        8192           15
             15
RA cache hit limit                        1         255            4
              4
RA cache miss limit                       1         255            3
              3
RA delay                                  0         500           15
             15
RA pre-fetches                            1        1000            2
              2
RA slots per thread                       1         255            5
              5
RA worker threads                         0         255            0
              0
recovery flags                            0           1            0
              0
recovery interval                         1       32767        32767
          32767
remote access                             0           1            0
              0
remote conn timeout                      -1       32767           10
             10
remote login timeout                      0  2147483647            5
              5
remote proc trans                         0           1            0
              0
remote query timeout                      0  2147483647            0
              0
remote sites                              0         256            0
              0
resource timeout                          5  2147483647           10
             10
set working set size                      0           1            1
              1
show advanced options                     0           1            1
              1
SMP concurrency                          -1          64           -1
             -1
sort pages                               64         511           64
             64
spin counter                              1  2147483647        10000
          10000
tempdb in ram (MB)                        0        2044            4
              4
time slice                               50        1000          100
            100
user connections                          5       32767          230
            230
user options                              0        4095            0
              0

(1 row affected)
1>
```

RAID Disk Controller Configuration

```
MegaRAID Ultra PCI Adapter (434 Rev. B)
BIOS Version 2.42 June 19, 1997

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

Host Adapter 1  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 1.

Logical Drive 1
State                : Optimal
RAID TYPE            : 0
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 32K Byte
No. of Stripes       : 7
No. of Spans         : 3
Size                 : 87129MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 0      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 1      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 1      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 2      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 2      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 3      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 1
        CHANNEL : 2, TARGET : 3      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 4      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 4      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 1, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 11     UNISYS   - 003557M2954E-512,

Capacity - 4149 MB
        CHANNEL : 2, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 13     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB

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

Host Adapter 2  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 1.

Logical Drive 1
State                : Optimal
RAID TYPE            : 0
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 32K Byte
No. of Stripes       : 5
No. of Spans         : 4
Size                 : 82980MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 0      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 1      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 1      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 2      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 1
        CHANNEL : 2, TARGET : 2      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 3      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 3      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 4      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 4      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 1, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
```

```
        CHANNEL : 2, TARGET : 9             UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 2, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB



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

Host Adapter 3  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 1.

Logical Drive 1
State                : Optimal
RAID TYPE            : 0
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 32K Byte
No. of Stripes       : 5
No. of Spans         : 4
Size                 : 128310MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 1
        CHANNEL : 2, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 2
        CHANNEL : 1, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 2, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB


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

Host Adapter 4  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 1.

Logical Drive 1
State                : Optimal
RAID TYPE            : 0
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 32K Byte
No. of Stripes       : 5
No. of Spans         : 4
Size                 : 128310MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 1
        CHANNEL : 2, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
```

```
        CHANNEL : 1, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 2
        CHANNEL : 1, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 2, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB



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

Host Adapter 5  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 1.

Logical Drive 1
State                : Optimal
RAID TYPE            : 0
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 32K Byte
No. of Stripes       : 5
No. of Spans         : 4
Size                 : 128310MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 1
        CHANNEL : 2, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 2
        CHANNEL : 1, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 2, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB

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

Host Adapter 6  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 1.

Logical Drive 1
State                : Optimal
RAID TYPE            : 0
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 32K Byte
No. of Stripes       : 5
No. of Spans        : 4
Size                 : 128310MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 1      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
```

```
        CHANNEL : 1, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 1
        CHANNEL : 2, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 1, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 2, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 2
        CHANNEL : 1, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 8      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 9      UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
RANK 2
        CHANNEL : 2, TARGET : 10     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 11     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 1, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB
        CHANNEL : 2, TARGET : 12     UNISYS   - 003557M2954E-512,
Capacity - 4149 MB



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

Host Adapter 7  Firmware Version Xm75  DRAM Size = 16MB

Number Of Logical Drives: 2.

Logical Drive 1
State                : Optimal
RAID TYPE            : 1
Write Policy         : Write Thru
Read Policy          : No Read Ahead
Cache Policy         : Direct I/O
Stripe Size          : 128K Byte
No. of Stripes       : 2
No. of Spans         : 4
Size                 : 34728MB
Component Physical Drives :
RANK 0
        CHANNEL : 1, TARGET : 0      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
        CHANNEL : 2, TARGET : 0      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
RANK 1
        CHANNEL : 1, TARGET : 1      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
        CHANNEL : 2, TARGET : 1      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
RANK 2
        CHANNEL : 1, TARGET : 8      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
        CHANNEL : 2, TARGET : 8      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
RANK 3
        CHANNEL : 1, TARGET : 9      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB
        CHANNEL : 2, TARGET : 9      UNISYS  - 006405ST19101W-512,
Capacity - 8683 MB

Logical Drive 2
State                : Optimal
RAID TYPE            : 5
Write Policy         : Write Thru
Read Policy          : Adaptive Read Ahead
Cache Policy         : Cached I/O
Stripe Size          : 128K Byte
No. of Stripes       : 6
No. of Spans         : 2
Size                 : 86820MB
Component Physical Drives :
RANK 0
        CHANNEL : 3, TARGET : 0      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 1      UNISYS   - 006405ST19101W-512,
Capacity - 8683 MB
        CHANNEL : 3, TARGET : 2      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 3      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 4      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 6      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
RANK 1
        CHANNEL : 3, TARGET : 8      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 9      UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 10     UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 11     UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 12     UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
        CHANNEL : 3, TARGET : 13     UNISYS   - 007434M2949E-512,
Capacity - 8682 MB
```

Configuration of Log Drives

There were four mirrored pairs of log drives, with four log drives housed in one independently powered disk cage and their four mirrors housed in a second independently powered disk cage. Each disk cage contained redundant power supplies. The two disk cages were attached to two channels of the seventh RAID controller. The controller was configured to stripe log IO across the four pairs of log drives and to use Write Thru, No Read Ahead and DirectIO (no read or write caching in the RAID controller).

The log disk drives themselves have large data buffers and support both Read Cache Enable (RCE, Factory enabled) and Write Cache Enable (WCE, Factory disabled). All disk writes first load data into the disk buffer. If WCE is enabled, IO completion is signaled immediately, and data is transferred to the media as soon as possible. If WCE is disabled, IO completion is not signaled until after the data is transferred to the media.

Unisys software (provided with the base system) was used to change the setting of WCE for each log drive to enabled.

For the priced configuration, a UPS was attached to the power supply of each of the two log disk cages. Since even with WCE enabled, the log drives will transfer the data from the buffer to the media as soon as possible (i.e., during the next disk revolution for sequential writes), the existence of the UPS guarantees that a loss of system power will not stop the writes from occurring to the disk media after IO completion is signaled to the RAID controller and system. Since there is no single point of failure in this configuration, it meets the TPC-C requirements of guaranteeing the durability of all committed transactions.

NT Server Configuration Information

```
Microsoft Diagnostics Report For \\HS6SUT
------------------------------------------------------------------------

OS Version Report
------------------------------------------------------------------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiprocessor Free
Registered Owner: SAM&M, Unisys Corporation
Product Number: 50382-123-1234567-69937
------------------------------------------------------------------------

System Report
------------------------------------------------------------------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 09/30/97
BIOS Version: PhoenixBIOS 4.0 Release 5.16.9B1

Processor list:
   0:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~200 Mhz
   1:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~200 Mhz
   2:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~200 Mhz
   3:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~200 Mhz
   4:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~200 Mhz
   5:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~200 Mhz
------------------------------------------------------------------------

Video Display Report
------------------------------------------------------------------------
BIOS Date: 09/02/94
BIOS Version: CL-GD5429 VGA BIOS Version 1.00a

Adapter:
   Setting: 1024 x 768 x 256
            72 Hz
   Type: cirrus compatible display adapter
   String: Cirrus Logic Compatible
   Memory: 1 MB
   Chip Type: CL 5429
   DAC Type: Integrated RAMDAC
Driver:
   Vendor: Microsoft Corporation
   File(s): cirrus.sys, vga.dll, cirrus.dll, vga256.dll, vga64K.dll
   Version: 4.00, 4.0.0


Drives Report
------------------------------------------------------------------------
C:\  (Local - NTFS) SYSTEM Total: 0KB, Free: 0KB
   Serial Number: E006 - C69
   Bytes per cluster: 512
   Sectors per cluster: 1
   Filename length: 255
D:\  (Local - NTFS) DUMP10 Total: 4,241,128KB, Free: 3,458,380KB
   Serial Number: C4B4 - 84B6
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
W:\  (Local - NTFS) DUMP1110 Total: 88,895,644KB, Free: 5,948,264KB
   Serial Number: 8B1 - 8407
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
X:\  (Local - NTFS) FILES Total: 2,152,708KB, Free: 738,076KB
   Serial Number: D0BB - 55B8
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255

Memory Report
------------------------------------------------------------------------
```

```
Handles: 1,220
Threads: 120
Processes: 15

Physical Memory (K)
   Total: 3,931,568
   Available: 1,058,428
   File Cache: 14,676

Kernel Memory (K)
   Total: 865,744
   Paged: 5,540
   Nonpaged: 860,204

Commit Charge (K)
   Total: 3,085,648
   Limit: 4,039,980
   Peak: 3,089,896

Pagefile Space (K)
   Total: 262,144
   Total in use: 3,004
   Peak: 3,200

   X:\pagefile.sys
      Total: 262,144
      Total in use: 3,004
      Peak: 3,200



Services Report
----------------------------------------------------------------------
Alerter                                    Running   (Automatic)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
Computer Browser                           Running   (Automatic)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
      LmHosts
ClipBook Server                            Stopped   (Manual)
   X:\WINNTG\system32\clipsrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      NetDDE
DHCP Client (TDI)                          Stopped   (Disabled)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Tcpip
      Afd
      NetBT
EventLog (Event log)                       Running   (Automatic)
   X:\WINNTG\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Server                                     Running   (Automatic)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
Workstation (NetworkProvider)              Running   (Automatic)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
License Logging Service                    Stopped   (Manual)
   X:\WINNTG\System32\llssrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TCP/IP NetBIOS Helper                      Running   (Automatic)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      NetworkProvider
Messenger                                  Stopped   (Manual)
   X:\WINNTG\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      NetBios
MSDTC (MS Transactions)                    Stopped   (Manual)
   C:\MSSQL\BINN\msdtc.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      RPCSS
MSSQLServer                                Stopped   (Manual)
   C:\MSSQL\BINN\SQLSERVR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
Network DDE (NetDDEGroup)                  Stopped   (Manual)
   X:\WINNTG\system32\netdde.exe
```

```
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      NetDDEDSDM
Network DDE DSDM                          Stopped   (Manual)
   X:\WINNTG\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Net Logon (RemoteValidation)              Stopped   (Manual)
   X:\WINNTG\System32\lsass.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LmHosts
NT LM Security Support Provider           Stopped   (Manual)
   X:\WINNTG\System32\SERVICES.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Plug and Play (PlugPlay)                  Stopped   (Manual)
   X:\WINNTG\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Directory Replicator                      Stopped   (Manual)
   X:\WINNTG\System32\lmrepl.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
Remote Procedure Call (RPC) Locator       Stopped   (Manual)
   X:\WINNTG\System32\LOCATOR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      Rdr
Remote Procedure Call (RPC) Service       Running   (Automatic)
   X:\WINNTG\system32\RpcSs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Schedule                                  Stopped   (Manual)
   X:\WINNTG\System32\AtSvc.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Simple TCP/IP Services                    Running   (Automatic)
   X:\WINNTG\system32\tcpsvcs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Afd
   Group Dependencies:
      TDI
SNMP                                      Running   (Automatic)
   X:\WINNTG\System32\snmp.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      Tcpip
      EventLog
SNMP Trap Service                         Stopped   (Manual)
   X:\WINNTG\System32\snmptrap.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      Tcpip
      EventLog
Spooler (SpoolerGroup)                    Stopped   (Manual)
   X:\WINNTG\system32\spoolss.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
SQLExecutive                              Stopped   (Manual)
   C:\MSSQL\BINN\SQLEXEC.EXE
   Service Account Name: .\sa
   Error Severity: Normal
   Service Flags: Own Process
Telephony Service                         Stopped   (Manual)
   X:\WINNTG\system32\tapisrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
UPS                                       Stopped   (Manual)
   X:\WINNTG\System32\ups.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process



Drivers Report
--------------------------------------------------------------------------
Abiosdsk (Primary disk)                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
AFD Networking Support Environment (TDI)  Running   (Automatic)
   X:\WINNTG\System32\drivers\afd.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha154x (SCSI miniport)                   Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha174x (SCSI miniport)                   Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
aic78xx (SCSI miniport)                   Running   (Boot)
```

```
   X:\WINNTG\System32\DRIVERS\aic78xx.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Always (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ami0nt (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
amsint (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Arrow (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Asante PCI Adapter Driver (NDIS)                Running   (Automatic)
   X:\WINNTG\System32\drivers\asantpci.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aspi32                                          Running   (Automatic)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
atapi (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Atdisk (Primary disk)                           Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ati (Video)                                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Beep (Base)                                     Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
BusLogic (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Busmouse (Pointer Port)                         Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdaudio (Filter)                                Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdfs (File system)                              Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      SCSI CDROM Class
Cdrom (SCSI CDROM Class)                        Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Changer (Filter)                                Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
cirrus (Video)                                  Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Cpqarray (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
cpqfws2e (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dac960nt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dce376nt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Delldsa (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Dell_DGX (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Disk (SCSI Class)                               Running   (Boot)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
diskint                                         Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Diskperf (Filter)                               Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
DptScsi (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dtc329x (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
em (Base)                                       Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
et4000 (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Fastfat (Boot file system)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Fd16_700 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd7000ex (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd8xx (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
flashpnt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Floppy (Primary disk)                           Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
```

```
Ftdisk (Filter)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running
(System)
   System32\DRIVERS\i8042prt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Inport (Pointer Port)                           Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg300 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg364 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jzvxl484 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Keyboard Class Driver (Keyboard Class)          Running   (System)
   System32\DRIVERS\kbdclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
KSecDD (Base)                                   Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mga (Video)                                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mga_mil (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mitsumi (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mkecr5xx (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Modem (Extended base)                           Stopped   (Manual)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Mouse Class Driver (Pointer Class)              Running   (System)
   System32\DRIVERS\mouclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mraid (Primary disk)                            Running   (Boot)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mraid35x (SCSI miniport)                        Stopped   (Boot)
   X:\WINNTG\system32\drivers\mraid35x.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Msfs (File system)                              Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                                   Running   (Manual)
   X:\WINNTG\System32\drivers\mup.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ncr53c9x (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ncr77c22 (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ncrc700 (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ncrc710 (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Microsoft NDIS System Driver (NDIS)             Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
NetBIOS Interface (NetBIOSGroup)                Stopped   (Manual)
   X:\WINNTG\System32\drivers\netbios.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      TDI
WINS Client(TCP/IP) (PNP_TDI)                   Stopped   (Automatic)
   X:\WINNTG\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                                       Stopped   (Manual)
   X:\WINNTG\system32\drivers\netdtect.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Npfs (File system)                              Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ntfs (File system)                              Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Null (Base)                                     Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
NWLink IPX/SPX Compatible Transport Protocol (PNP_TDI) Running
(Automatic)
   X:\WINNTG\System32\drivers\nwlnkipx.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
NWLink NetBIOS (PNP_TDI)                        Running   (Automatic)
   X:\WINNTG\System32\drivers\nwlnknb.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      NwlnkIpx
NWLink SPX/SPXII Protocol                       Running   (Manual)
   X:\WINNTG\System32\drivers\nwlnkspx.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      NwlnkIpx
Oliscsi (SCSI miniport)                         Stopped   (Disabled)
```

```
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Parallel (Extended base)                    Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
Parport (Parallel arbitrator)               Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ParVdm (Extended base)                      Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
PCIDump (PCI Configuration)                 Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Pcmcia (System Bus Extender)                Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
PnP ISA Enabler Driver (Base)               Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
PortFltr (port)                             Stopped   (Manual)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
psidisp (Video)                             Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ql10wnt (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
qv (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Rdr (Network)                               Running   (Manual)
   X:\WINNTG\System32\drivers\rdr.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
s3 (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Scsiprnt (Extended base)                    Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Scsiscan (SCSI Class)                       Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Serial (Extended base)                      Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sermouse (Pointer Port)                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sfloppy (Primary disk)                      Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Simbad (Filter)                             Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
slcd32 (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Sparrow (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Spock (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Srv (Network)                               Running   (Manual)
   X:\WINNTG\System32\drivers\srv.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
symc810 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T128 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T13B (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
TCP/IP Service (PNP_TDI)                    Running   (Automatic)
   X:\WINNTG\System32\drivers\tcpip.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
tga (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
tmv1 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra124 (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra14f (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra24f (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
v7vram (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaSave (Video Save)                        Stopped   (System)
```

```
   X:\WINNTG\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                          Stopped   (System)
   X:\WINNTG\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Wd33c93 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
wd90c24a (Video)                               Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
wdvga (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
weitekp9 (Video)                               Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Xga (Video)                                    Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
```

IRQ and Port Report

Devices	Vector	Level	Affinity
MPS 1.4 - APIC platform	8	8	0x0000003f
MPS 1.4 - APIC platform	0	0	0x0000003f
MPS 1.4 - APIC platform	1	1	0x0000003f
MPS 1.4 - APIC platform	2	2	0x0000003f
MPS 1.4 - APIC platform	3	3	0x0000003f
MPS 1.4 - APIC platform	4	4	0x0000003f
MPS 1.4 - APIC platform	5	5	0x0000003f
MPS 1.4 - APIC platform	6	6	0x0000003f
MPS 1.4 - APIC platform	7	7	0x0000003f
MPS 1.4 - APIC platform	8	8	0x0000003f
MPS 1.4 - APIC platform	9	9	0x0000003f
MPS 1.4 - APIC platform	10	10	0x0000003f
MPS 1.4 - APIC platform	11	11	0x0000003f
MPS 1.4 - APIC platform	12	12	0x0000003f
MPS 1.4 - APIC platform	13	13	0x0000003f
MPS 1.4 - APIC platform	14	14	0x0000003f
MPS 1.4 - APIC platform	15	15	0x0000003f
MPS 1.4 - APIC platform	16	16	0x0000003f
MPS 1.4 - APIC platform	17	17	0x0000003f
MPS 1.4 - APIC platform	18	18	0x0000003f
MPS 1.4 - APIC platform	19	19	0x0000003f
MPS 1.4 - APIC platform	20	20	0x0000003f
MPS 1.4 - APIC platform	21	21	0x0000003f
MPS 1.4 - APIC platform	22	22	0x0000003f
MPS 1.4 - APIC platform	23	23	0x0000003f
MPS 1.4 - APIC platform	24	24	0x0000003f
MPS 1.4 - APIC platform	25	25	0x0000003f
MPS 1.4 - APIC platform	26	26	0x0000003f
MPS 1.4 - APIC platform	27	27	0x0000003f
MPS 1.4 - APIC platform	28	28	0x0000003f
MPS 1.4 - APIC platform	29	29	0x0000003f
MPS 1.4 - APIC platform	30	30	0x0000003f
MPS 1.4 - APIC platform	31	31	0x0000003f
MPS 1.4 - APIC platform	32	32	0x0000003f
MPS 1.4 - APIC platform	33	33	0x0000003f
MPS 1.4 - APIC platform	34	34	0x0000003f
MPS 1.4 - APIC platform	35	35	0x0000003f
MPS 1.4 - APIC platform	36	36	0x0000003f
MPS 1.4 - APIC platform	37	37	0x0000003f
MPS 1.4 - APIC platform	38	38	0x0000003f
MPS 1.4 - APIC platform	39	39	0x0000003f
MPS 1.4 - APIC platform	40	40	0x0000003f
MPS 1.4 - APIC platform	41	41	0x0000003f
MPS 1.4 - APIC platform	42	42	0x0000003f
MPS 1.4 - APIC platform	43	43	0x0000003f
MPS 1.4 - APIC platform	44	44	0x0000003f
MPS 1.4 - APIC platform	45	45	0x0000003f
MPS 1.4 - APIC platform	46	46	0x0000003f
MPS 1.4 - APIC platform	47	47	0x0000003f
MPS 1.4 - APIC platform	61	61	0x0000003f
MPS 1.4 - APIC platform	65	65	0x0000003f
MPS 1.4 - APIC platform	80	80	0x0000003f
MPS 1.4 - APIC platform	193	193	0x0000003f
MPS 1.4 - APIC platform	225	225	0x0000003f
MPS 1.4 - APIC platform	253	253	0x0000003f
MPS 1.4 - APIC platform	254	254	0x0000003f
MPS 1.4 - APIC platform	255	255	0x0000003f
i8042prt	1	1	0xffffffff
i8042prt	12	12	0xffffffff
Serial	4	4	0x00000000
Serial	3	3	0x00000000
AsantePCI	10	10	0x00000000
Floppy	6	6	0x00000000
aic78xx	10	10	0x00000000
mraid	14	14	0x00000000
mraid	11	11	0x00000000
mraid	15	15	0x00000000
mraid	10	10	0x00000000
mraid	14	14	0x00000000
mraid	11	11	0x00000000
mraid	15	15	0x00000000

Devices	Physical Address	Length
MPS 1.4 - APIC platform	0x00000000	0x0000000010
MPS 1.4 - APIC platform	0x00000020	0x0000000002
MPS 1.4 - APIC platform	0x00000040	0x0000000004
MPS 1.4 - APIC platform	0x00000048	0x0000000004
MPS 1.4 - APIC platform	0x00000061	0x0000000001
MPS 1.4 - APIC platform	0x00000070	0x0000000002
MPS 1.4 - APIC platform	0x00000080	0x0000000010
MPS 1.4 - APIC platform	0x00000092	0x0000000001
MPS 1.4 - APIC platform	0x000000a0	0x0000000002
MPS 1.4 - APIC platform	0x000000c0	0x0000000010
MPS 1.4 - APIC platform	0x000000d0	0x0000000010
MPS 1.4 - APIC platform	0x000000f0	0x0000000010
MPS 1.4 - APIC platform	0x00000400	0x0000000010
MPS 1.4 - APIC platform	0x00000461	0x0000000002
MPS 1.4 - APIC platform	0x00000464	0x0000000002

```
MPS 1.4 - APIC platform         0x00000480  0x0000000010
MPS 1.4 - APIC platform         0x000004c2  0x000000000e
MPS 1.4 - APIC platform         0x000004d0  0x0000000002
MPS 1.4 - APIC platform         0x000004d4  0x000000002c
MPS 1.4 - APIC platform         0x00000c84  0x0000000001
i8042prt                        0x00000060  0x0000000001
i8042prt                        0x00000064  0x0000000001
Parport                         0x00000378  0x0000000003
Serial                          0x000003f8  0x0000000007
Serial                          0x000002f8  0x0000000007
AsantePCI                       0x0000f880  0x0000000080
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
aic78xx                         0x0000f000  0x0000000100
mraid                           0x0000f480  0x0000000080
mraid                           0x0000fc00  0x0000000080
mraid                           0x0000f800  0x0000000080
mraid                           0x0000ec00  0x0000000080
mraid                           0x0000e880  0x0000000080
mraid                           0x0000e800  0x0000000080
mraid                           0x0000e480  0x0000000080
cirrus                          0x000003b0  0x000000000c
cirrus                          0x000003c0  0x0000000020



DMA and Memory Report
-------------------------------------------------------------------------
Devices                         Channel    Port
-------------------------------------------------------------------------
Floppy                             2     0
-------------------------------------------------------------------------
Devices                         Physical Address  Length
-------------------------------------------------------------------------
MPS 1.4 - APIC platform         0xfec00000  0x00000400
MPS 1.4 - APIC platform         0xfee00000  0x00000400
aic78xx                         0xfe9fe000  0x00001000
cirrus                          0x000a0000  0x00020000



Environment Report
-------------------------------------------------------------------------



System Environment Variables
   ComSpec=X:\WINNTG\system32\cmd.exe
   HOME=X:/
   NUMBER_OF_PROCESSORS=6
   OS=Windows_NT
   Os2LibPath=X:\WINNTG\system32\os2\dll;

Path=X:\MKSDemo\mksnt;X:\WINNTG\system32;X:\WINNTG;;C:\MSSQL\BINN;x:\intel
\emon\emon\bin
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 1 Stepping 9, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=0109
   ROOTDIR=X:/MKSDemo
   SHELL=X:/MKSDemo/mksnt/sh.exe
   TMPDIR=X:/TEMP
   windir=X:\WINNTG



Environment Variables for Current User
   TEMP=X:\TEMP
   TMP=X:\TEMP



Network Report
-------------------------------------------------------------------------
Your Access Level: Admin & Local
Workgroup or Domain: WORKGROUP
Network Version: 4.0
LanRoot: WORKGROUP
Logged On Users: 1
Current User (1): Administrator
  Logon Domain: HS6SUT
  Logon Server: HS6SUT

Transport: NwlnkNb, 00-00-94-79-6D-68, VC's: 2, Wan: Wan

Character Wait: 3,600
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 50
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
Log Election Packets: False
Use Opportunistic Locking: True
Use Unlock Behind: True
Use Close Behind: True
Buffer Pipes: True
Use Lock, Read, Unlock: True
Use NT Caching: True
Use Raw Read: True
Use Raw Write: True
Use Write Raw Data: True
Use Encryption: True
Buffer Deny Write Files: True
Buffer Read Only Files: True
```

```
Force Core Creation: True
512 Byte Max Transfer: False
Bytes Received: 24,569,653
SMB's Received: 149,692
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 711,306,240
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 355,653,120
Bytes Transmitted: 10,748,616
SMB's Transmitted: 149,692
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 1,169,003
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Initally Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 86,832
Random Read Operations: 86,825
Read SMB's: 86,832
Large Read SMB's: 0
Small Read SMB's: 3
Write Operations: 37,247
Random Write Operations: 0
Write SMB's: 37,246
Large Write SMB's: 0
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 80
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 79
Server Disconnects: 1
Hung Sessions: 0
Use Count: 156
Failed Use Count: 0
Current Commands: 0
Server File Opens: 3
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 1
Server Sessions Timed Out: 0
Server Sessions Errored Out: 0
Server Password Errors: 0
Server Permission Errors: 0
Server System Errors: 0
Server Bytes Sent: 24,523,912
Server Bytes Received: 10,696,166
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

Internet Information Server Registry Parameters

```
Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM

Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
Class Name:        <NO CLASS>
Last Write Time:   10/3/97 - 10:22 AM
Value 0
  Name:            BandwidthLevel
  Type:            REG_DWORD
  Data:            0xffffffff

Value 1
  Name:            ListenBackLog
  Type:            REG_DWORD
  Data:            0x19

Value 2
  Name:            PoolThreadLimit
  Type:            REG_DWORD
  Data:            0x64


Key Name:
SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\Filter
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            FilterType
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            NumDenySites
  Type:            REG_DWORD
  Data:            0

Value 2
  Name:            NumGrantSites
  Type:            REG_DWORD
  Data:            0


Key Name:
SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\MimeMap
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            application/envoy,evy,,5
  Type:            REG_SZ
  Data:

Value 1
  Name:            application/mac-binhex40,hqx,,4
  Type:            REG_SZ
  Data:

Value 2
  Name:            application/msword,doc,,5
  Type:            REG_SZ
  Data:

Value 3
  Name:            application/msword,dot,,5
  Type:            REG_SZ
  Data:

Value 4
  Name:            application/octet-stream,*,,5
  Type:            REG_SZ
  Data:

Value 5
  Name:            application/octet-stream,bin,,5
  Type:            REG_SZ
  Data:

Value 6
  Name:            application/octet-stream,exe,,5
  Type:            REG_SZ
  Data:

Value 7
  Name:            application/oda,oda,,5
  Type:            REG_SZ
  Data:

Value 8
  Name:            application/pdf,pdf,,5
  Type:            REG_SZ
  Data:

Value 9
  Name:            application/postscript,ai,,5
  Type:            REG_SZ
  Data:

Value 10
  Name:            application/postscript,eps,,5
  Type:            REG_SZ
  Data:

Value 11
  Name:            application/postscript,ps,,5
  Type:            REG_SZ
  Data:

Value 12
  Name:            application/rtf,rtf,,5
  Type:            REG_SZ
  Data:
```

```
Value 13
  Name:           application/winhlp,hlp,,5
  Type:           REG_SZ
  Data:

Value 14
  Name:           application/x-bcpio,bcpio,,5
  Type:           REG_SZ
  Data:

Value 15
  Name:           application/x-cpio,cpio,,5
  Type:           REG_SZ
  Data:

Value 16
  Name:           application/x-csh,csh,,5
  Type:           REG_SZ
  Data:

Value 17
  Name:           application/x-director,dcr,,5
  Type:           REG_SZ
  Data:

Value 18
  Name:           application/x-director,dir,,5
  Type:           REG_SZ
  Data:

Value 19
  Name:           application/x-director,dxr,,5
  Type:           REG_SZ
  Data:

Value 20
  Name:           application/x-dvi,dvi,,5
  Type:           REG_SZ
  Data:

Value 21
  Name:           application/x-gtar,gtar,,9
  Type:           REG_SZ
  Data:

Value 22
  Name:           application/x-hdf,hdf,,5
  Type:           REG_SZ
  Data:

Value 23
  Name:           application/x-latex,latex,,5
  Type:           REG_SZ
  Data:

Value 24
  Name:           application/x-msaccess,mdb,,5
  Type:           REG_SZ
  Data:

Value 25
  Name:           application/x-mscardfile,crd,,5
  Type:           REG_SZ
  Data:

Value 26
  Name:           application/x-msclip,clp,,5
  Type:           REG_SZ
  Data:

Value 27
  Name:           application/x-msexcel,xla,,5
  Type:           REG_SZ
  Data:

Value 28
  Name:           application/x-msexcel,xlc,,5
  Type:           REG_SZ
  Data:

Value 29
  Name:           application/x-msexcel,xlm,,5
  Type:           REG_SZ
  Data:

Value 30
  Name:           application/x-msexcel,xls,,5
  Type:           REG_SZ
  Data:

Value 31
  Name:           application/x-msexcel,xlt,,5
  Type:           REG_SZ
  Data:

Value 32
  Name:           application/x-msexcel,xlw,,5
  Type:           REG_SZ
  Data:

Value 33
  Name:           application/x-msmediaview,m13,,5
  Type:           REG_SZ
  Data:

Value 34
  Name:           application/x-msmediaview,m14,,5
  Type:           REG_SZ
  Data:

Value 35
  Name:           application/x-msmetafile,wmf,,5
  Type:           REG_SZ
  Data:

Value 36
  Name:           application/x-msmoney,mny,,5
  Type:           REG_SZ
```

```
  Data:

Value 37
  Name:           application/x-mspowerpoint,ppt,,5
  Type:           REG_SZ
  Data:

Value 38
  Name:           application/x-msproject,mpp,,5
  Type:           REG_SZ
  Data:

Value 39
  Name:           application/x-mspublisher,pub,,5
  Type:           REG_SZ
  Data:

Value 40
  Name:           application/x-msterminal,trm,,5
  Type:           REG_SZ
  Data:

Value 41
  Name:           application/x-msworks,wks,,5
  Type:           REG_SZ
  Data:

Value 42
  Name:           application/x-mswrite,wri,,5
  Type:           REG_SZ
  Data:

Value 43
  Name:           application/x-netcdf,cdf,,5
  Type:           REG_SZ
  Data:

Value 44
  Name:           application/x-netcdf,nc,,5
  Type:           REG_SZ
  Data:

Value 45
  Name:           application/x-perfmon,pma,,5
  Type:           REG_SZ
  Data:

Value 46
  Name:           application/x-perfmon,pmc,,5
  Type:           REG_SZ
  Data:

Value 47
  Name:           application/x-perfmon,pml,,5
  Type:           REG_SZ
  Data:

Value 48
  Name:           application/x-perfmon,pmr,,5
  Type:           REG_SZ
  Data:

Value 49
  Name:           application/x-perfmon,pmw,,5
  Type:           REG_SZ
  Data:

Value 50
  Name:           application/x-sh,sh,,5
  Type:           REG_SZ
  Data:

Value 51
  Name:           application/x-shar,shar,,5
  Type:           REG_SZ
  Data:

Value 52
  Name:           application/x-sv4cpio,sv4cpio,,5
  Type:           REG_SZ
  Data:

Value 53
  Name:           application/x-sv4crc,sv4crc,,5
  Type:           REG_SZ
  Data:

Value 54
  Name:           application/x-tar,tar,,5
  Type:           REG_SZ
  Data:

Value 55
  Name:           application/x-tcl,tcl,,5
  Type:           REG_SZ
  Data:

Value 56
  Name:           application/x-tex,tex,,5
  Type:           REG_SZ
  Data:

Value 57
  Name:           application/x-texinfo,texi,,5
  Type:           REG_SZ
  Data:

Value 58
  Name:           application/x-texinfo,texinfo,,5
  Type:           REG_SZ
  Data:

Value 59
  Name:           application/x-troff,roff,,5
  Type:           REG_SZ
  Data:

Value 60
```

```
  Name:           application/x-troff,t,,5
  Type:           REG_SZ
  Data:

Value 61
  Name:           application/x-troff,tr,,5
  Type:           REG_SZ
  Data:

Value 62
  Name:           application/x-troff-man,man,,5
  Type:           REG_SZ
  Data:

Value 63
  Name:           application/x-troff-me,me,,5
  Type:           REG_SZ
  Data:

Value 64
  Name:           application/x-troff-ms,ms,,5
  Type:           REG_SZ
  Data:

Value 65
  Name:           application/x-ustar,ustar,,5
  Type:           REG_SZ
  Data:

Value 66
  Name:           application/x-wais-source,src,,7
  Type:           REG_SZ
  Data:

Value 67
  Name:           application/zip,zip,,9
  Type:           REG_SZ
  Data:

Value 68
  Name:           audio/basic,au,,<
  Type:           REG_SZ
  Data:

Value 69
  Name:           audio/basic,snd,,<
  Type:           REG_SZ
  Data:

Value 70
  Name:           audio/x-aiff,aif,,<
  Type:           REG_SZ
  Data:

Value 71
  Name:           audio/x-aiff,aifc,,<
  Type:           REG_SZ
  Data:

Value 72
  Name:           audio/x-aiff,aiff,,<
  Type:           REG_SZ
  Data:

Value 73
  Name:           audio/x-pn-realaudio,ram,,<
  Type:           REG_SZ
  Data:

Value 74
  Name:           audio/x-wav,wav,,<
  Type:           REG_SZ
  Data:

Value 75
  Name:           image/bmp,bmp,,:
  Type:           REG_SZ
  Data:

Value 76
  Name:           image/cis-cod,cod,,5
  Type:           REG_SZ
  Data:

Value 77
  Name:           image/gif,gif,,g
  Type:           REG_SZ
  Data:

Value 78
  Name:           image/ief,ief,,:
  Type:           REG_SZ
  Data:

Value 79
  Name:           image/jpeg,jpe,,:
  Type:           REG_SZ
  Data:

Value 80
  Name:           image/jpeg,jpeg,,:
  Type:           REG_SZ
  Data:

Value 81
  Name:           image/jpeg,jpg,,:
  Type:           REG_SZ
  Data:

Value 82
  Name:           image/tiff,tif,,:
  Type:           REG_SZ
  Data:

Value 83
  Name:           image/tiff,tiff,,:
  Type:           REG_SZ
  Data:
```

```
Value 84
  Name:           image/x-cmu-raster,ras,,:
  Type:           REG_SZ
  Data:

Value 85
  Name:           image/x-cmx,cmx,,5
  Type:           REG_SZ
  Data:

Value 86
  Name:           image/x-portable-anymap,pnm,,:
  Type:           REG_SZ
  Data:

Value 87
  Name:           image/x-portable-bitmap,pbm,,:
  Type:           REG_SZ
  Data:

Value 88
  Name:           image/x-portable-graymap,pgm,,:
  Type:           REG_SZ
  Data:

Value 89
  Name:           image/x-portable-pixmap,ppm,,:
  Type:           REG_SZ
  Data:

Value 90
  Name:           image/x-rgb,rgb,,:
  Type:           REG_SZ
  Data:

Value 91
  Name:           image/x-xbitmap,xbm,,:
  Type:           REG_SZ
  Data:

Value 92
  Name:           image/x-xpixmap,xpm,,:
  Type:           REG_SZ
  Data:

Value 93
  Name:           image/x-xwindowdump,xwd,,:
  Type:           REG_SZ
  Data:

Value 94
  Name:           text/html,htm,,h
  Type:           REG_SZ
  Data:

Value 95
  Name:           text/html,html,,h
  Type:           REG_SZ
  Data:

Value 96
  Name:           text/html,stm,,h
  Type:           REG_SZ
  Data:

Value 97
  Name:           text/plain,bas,,0
  Type:           REG_SZ
  Data:

Value 98
  Name:           text/plain,c,,0
  Type:           REG_SZ
  Data:

Value 99
  Name:           text/plain,h,,0
  Type:           REG_SZ
  Data:

Value 100
  Name:           text/plain,txt,,0
  Type:           REG_SZ
  Data:

Value 101
  Name:           text/richtext,rtx,,0
  Type:           REG_SZ
  Data:

Value 102
  Name:           text/tab-separated-values,tsv,,0
  Type:           REG_SZ
  Data:

Value 103
  Name:           text/x-setext,etx,,0
  Type:           REG_SZ
  Data:

Value 104
  Name:           video/mpeg,mpe,,;
  Type:           REG_SZ
  Data:

Value 105
  Name:           video/mpeg,mpeg,,;
  Type:           REG_SZ
  Data:

Value 106
  Name:           video/mpeg,mpg,,;
  Type:           REG_SZ
  Data:

Value 107
  Name:           video/quicktime,mov,,;
```

```
  Type:            REG_SZ
  Data:

Value 108
  Name:            video/quicktime,qt,,;
  Type:            REG_SZ
  Data:

Value 109
  Name:            video/x-msvideo,avi,,<
  Type:            REG_SZ
  Data:

Value 110
  Name:            video/x-sgi-movie,movie,,<
  Type:            REG_SZ
  Data:

Value 111
  Name:            x-world/x-vrml,flr,,5
  Type:            REG_SZ
  Data:

Value 112
  Name:            x-world/x-vrml,wrl,,5
  Type:            REG_SZ
  Data:

Value 113
  Name:            x-world/x-vrml,wrz,,5
  Type:            REG_SZ
  Data:

Value 114
  Name:            x-world/x-vrml,xaf,,5
  Type:            REG_SZ
  Data:

Value 115
  Name:            x-world/x-vrml,xof,,5
  Type:            REG_SZ
  Data:


Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Performance
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            Close
  Type:            REG_SZ
  Data:            CloseINFOPerformanceData

Value 1
  Name:            Collect
  Type:            REG_SZ
  Data:            CollectINFOPerformanceData

Value 2
  Name:            First Counter
  Type:            REG_DWORD
  Data:            0x7e6

Value 3
  Name:            First Help
  Type:            REG_DWORD
  Data:            0x7e7

Value 4
  Name:            Last Counter
  Type:            REG_DWORD
  Data:            0x804

Value 5
  Name:            Last Help
  Type:            REG_DWORD
  Data:            0x805

Value 6
  Name:            Library
  Type:            REG_SZ
  Data:            infoctrs.DLL

Value 7
  Name:            Open
  Type:            REG_SZ
  Data:            OpenINFOPerformanceData


Key Name:          SOFTWARE\Microsoft\Inetsrv
Class Name:        GenericClass
Last Write Time:   9/11/97 - 6:38 AM

Key Name:          SOFTWARE\Microsoft\Inetsrv\CurrentVersion
Class Name:        GenericClass
Last Write Time:   9/11/97 - 6:43 AM
Value 0
  Name:            Description
  Type:            REG_SZ
  Data:            Microsoft Internet Information Server 3.0

Value 1
  Name:            InstallDate
  Type:            REG_DWORD
  Data:            0x3417f43e

Value 2
  Name:            MajorVersion
  Type:            REG_DWORD
  Data:            0x4

Value 3
  Name:            MinorVersion
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            OperationsSupport
  Type:            REG_DWORD
```

```
  Data:            0x86

Value 5
  Name:            ServiceName
  Type:            REG_SZ
  Data:            Microsoft Internet Information Server 3.0

Value 6
  Name:            SoftwareType
  Type:            REG_SZ
  Data:            service

Value 7
  Name:            Title
  Type:            REG_SZ
  Data:            Microsoft Internet Information Server 3.0


Key Name:          SOFTWARE\Microsoft\Inetsrv\CurrentVersion\NetRules
Class Name:        GenericClass
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            InfName
  Type:            REG_SZ
  Data:            oemnsvin.inf

Value 1
  Name:            InfOption
  Type:            REG_SZ
  Data:            Inetsrv
```

World Wide Web Server Registry Parameters

```
Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC
Class Name:        <NO CLASS>
Last Write Time:   9/12/97 - 8:21 AM
Value 0
  Name:            DependOnGroup
  Type:            REG_MULTI_SZ
  Data:

Value 1
  Name:            DependOnService
  Type:            REG_MULTI_SZ
  Data:            RPCSS
                   NTLMSSP


Value 2
  Name:            DisplayName
  Type:            REG_SZ
  Data:            World Wide Web Publishing Service

Value 3
  Name:            ErrorControl
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            ImagePath
  Type:            REG_EXPAND_SZ
  Data:            C:\WINNT\System32\inetsrv\inetinfo.exe

Value 5
  Name:            ObjectName
  Type:            REG_SZ
  Data:            LocalSystem

Value 6
  Name:            Start
  Type:            REG_DWORD
  Data:            0x3

Value 7
  Name:            Type
  Type:            REG_DWORD
  Data:            0x20


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Enum
Class Name:        <NO CLASS>
Last Write Time:   10/6/97 - 4:05 PM
Value 0
  Name:            0
  Type:            REG_SZ
  Data:            Root\LEGACY_W3SVC\0000

Value 1
  Name:            Count
  Type:            REG_DWORD
  Data:            0x1

Value 2
  Name:            NextInstance
  Type:            REG_DWORD
  Data:            0x1


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 7:07 AM
Value 0
  Name:            AccessDeniedMessage
  Type:            REG_SZ
  Data:            Error: Access is Denied.

Value 1
  Name:            AdminEmail
  Type:            REG_SZ
  Data:            Admin@corp.com

Value 2
  Name:            AdminName
  Type:            REG_SZ
  Data:            Administrator
```

```
Value 3
  Name:           AnonymousUserName
  Type:           REG_SZ
  Data:           IUSR_XR6-CLIENT1

Value 4
  Name:           Authorization
  Type:           REG_DWORD
  Data:           0x5

Value 5
  Name:           CacheExtensions
  Type:           REG_DWORD
  Data:           0x1

Value 6
  Name:           CheckForWAISDB
  Type:           REG_DWORD
  Data:           0

Value 7
  Name:           ConnectionTimeOut
  Type:           REG_DWORD
  Data:           0x1c20

Value 8
  Name:           DebugFlags
  Type:           REG_DWORD
  Data:           0x8

Value 9
  Name:           Default Load File
  Type:           REG_SZ
  Data:           Default.htm

Value 10
  Name:           Dir Browse Control
  Type:           REG_DWORD
  Data:           0x4000001e

Value 11
  Name:           Filter DLLs
  Type:           REG_SZ
  Data:           C:\WINNT\System32\inetsrv\sspifilt.dll

Value 12
  Name:           GlobalExpire
  Type:           REG_DWORD
  Data:           0xffffffff

Value 13
  Name:           InstallPath
  Type:           REG_SZ
  Data:           C:\WINNT\System32\inetsrv

Value 14
  Name:           LogFileDirectory
  Type:           REG_EXPAND_SZ
  Data:           %SystemRoot%\System32\LogFiles

Value 15
  Name:           LogFileFormat
  Type:           REG_DWORD
  Data:           0

Value 16
  Name:           LogFilePeriod
  Type:           REG_DWORD
  Data:           0x1

Value 17
  Name:           LogFileTruncateSize
  Type:           REG_DWORD
  Data:           0x1388000

Value 18
  Name:           LogSqlDataSource
  Type:           REG_SZ
  Data:           HTTPLOG

Value 19
  Name:           LogSqlPassword
  Type:           REG_SZ
  Data:           sqllog

Value 20
  Name:           LogSqlTableName
  Type:           REG_SZ
  Data:           Internetlog

Value 21
  Name:           LogSqlUserName
  Type:           REG_SZ
  Data:           InternetAdmin

Value 22
  Name:           LogType
  Type:           REG_DWORD
  Data:           0

Value 23
  Name:           MajorVersion
  Type:           REG_DWORD
  Data:           0x2

Value 24
  Name:           MaxConnections
  Type:           REG_DWORD
  Data:           0x2710

Value 25
  Name:           MinorVersion
  Type:           REG_DWORD
  Data:           0

Value 26
  Name:           NTAuthenticationProviders
```

```
  Type:            REG_SZ
  Data:            NTLM

Value 27
  Name:            ScriptTimeout
  Type:            REG_DWORD
  Data:            0x384

Value 28
  Name:            SecurePort
  Type:            REG_DWORD
  Data:            0x1bb

Value 29
  Name:            ServerComment
  Type:            REG_SZ
  Data:

Value 30
  Name:            ServerSideIncludesEnabled
  Type:            REG_DWORD
  Data:            0x1

Value 31
  Name:            ServerSideIncludesExtension
  Type:            REG_SZ
  Data:            .stm


Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Script Map
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            .idc
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\httpodbc.dll


Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 7:07 AM
Value 0
  Name:            /,
  Type:            REG_SZ
  Data:            C:\InetPub\wwwroot,,5

Value 1
  Name:            /iisadmin,
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\iisadmin,,1

Value 2
  Name:            /Scripts,
  Type:            REG_SZ
  Data:            C:\InetPub\scripts,,4
```

```
Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Performance
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            Close
  Type:            REG_SZ
  Data:            CloseW3PerformanceData

Value 1
  Name:            Collect
  Type:            REG_SZ
  Data:            CollectW3PerformanceData

Value 2
  Name:            First Counter
  Type:            REG_DWORD
  Data:            0x806

Value 3
  Name:            First Help
  Type:            REG_DWORD
  Data:            0x807

Value 4
  Name:            Last Counter
  Type:            REG_DWORD
  Data:            0x83e

Value 5
  Name:            Last Help
  Type:            REG_DWORD
  Data:            0x83f

Value 6
  Name:            Library
  Type:            REG_SZ
  Data:            w3ctrs.DLL

Value 7
  Name:            Open
  Type:            REG_SZ
  Data:            OpenW3PerformanceData


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Security
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:38 AM
Value 0
  Name:            Security
  Type:            REG_BINARY
  Data:
00000000   01 00 14 80 c0 00 00 00 - cc 00 00 00 14 00 00 00
................
00000010   34 00 00 00 02 00 20 00 - 01 00 00 00 02 80 18 00  4.....
........
00000020   ff 01 0f 00 01 01 00 00 - 00 00 00 01 00 00 00 00
................
00000030   20 02 00 00 02 00 8c 00 - 05 00 00 00 00 00 18 00
...............
```

```
00000040   8d 01 02 00 01 01 00 00 - 00 00 00 01 00 00 00 00
................
00000050   ff ff ff ff 00 00 1c 00 - fd 01 02 00 01 02 00 00
................
00000060   00 00 00 05 20 00 00 00 - 23 02 00 00 08 b8 14 00  ....
...#.......
00000070   00 00 1c 00 ff 01 0f 00 - 01 02 00 00 00 00 00 05
................
00000080   20 00 00 00 20 02 00 00 - 08 b8 14 00 00 00 1c 00   ...
...........
00000090   ff 01 0f 00 01 02 00 00 - 00 00 00 05 20 00 00 00  ............
...
000000a0   25 02 00 00 08 b8 14 00 - 00 00 18 00 fd 01 02 00
%...............
000000b0   01 01 00 00 00 00 00 05 - 12 00 00 00 25 02 00 00
............%...
000000c0   01 01 00 00 00 00 00 05 - 12 00 00 00 01 01 00 00
................
000000d0   00 00 00 05 12 00 00 00 -                          ........


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\W3SAMP
Class Name:        <NO CLASS>
Last Write Time:   9/11/97 - 6:39 AM
```

Tuxedo Configuration

Note: this configuration file is repeated on each of the other 5 clients with the exception of the Hostname, "XR6-CLIENT1", which is replaced by "XR6-CLIENT2" thru "XR6-CLIENT6".

```
*RESOURCES
IPCKEY          133133

MAXACCESSERS    1900
MAXSERVERS      210
MAXSERVICES     1800
MODEL           SHM
MASTER          tpcctm
LDBAL           N
SCANUNIT        15
BLOCKTIME       60
BBLQUERY        60

*MACHINES
DEFAULT:

"XR6-CLIENT1"   LMID=tpcctm
                TUXDIR="c:\tuxedo"
                APPDIR="c:\tuxedo\runtime"
                TUXCONFIG="c:\tuxedo\runtime\tuxconfig"
                ULOGPFX="c:\tuxedo\runtime\ulog\ULOG"
                TYPE="WinNT"
                UID=0
                GID=0

*GROUPS
GROUPNT
        LMID=tpcctm     GRPNO=1   OPENINFO=NONE

*SERVERS
DEFAULT:
                CLOPT="-A -- -sHS6SUT -dtpcc"

tpccsvr         SRVGRP=GROUPNT
                SRVID=100
                MIN=35 MAX=200
                RQADDR=tpccq REPLYQ=Y

tpccdelv        SRVGRP=GROUPNT
                SRVID=301
                MIN=1 MAX=1
                CLOPT="-A -- -sHS6SUT -dtpcc -n301"
                RQADDR=delivery REPLYQ=Y

tpccdelv        SRVGRP=GROUPNT
                SRVID=302
                MIN=1 MAX=1
                CLOPT="-A -- -sHS6SUT -dtpcc -n302"
                RQADDR=delivery REPLYQ=Y

*SERVICES
```

Appendix D - RTE Code

Admin Environment

```
set WEBADMINCFG=web1065-24x6.cfg
set WEBMAXDRIVERS=24
set WEBDIAGLEVEL=4
set WEBEVENTLOG=0
set WEBEVENTHOST=
set WEBCHECKLEVEL=2

webadmin.exe
```

Profiles used for Performance Run

web1065-24x6.cfg

```
//
//  Common Driver Configuration
//
INITBASEPORT 4300
INITSYNCMAX 4
INITPAUSE 1
INITRSCALE 220
INITTSCALE 100
INITRWID 1, 1065
INITFIXEDWID 1
INITCCLAST 223
INITCCID 223
INITCITEMID 223
//
//  Configuration Driver 1
//
1 INITIPADDR 192.59.13.228
1 INITIISADDR 192.168.83.1
1 INITIISPORT 80
1 INITBROWSERS 440
1 INITMYWID 1,44
//
//  Configuration Driver 2
//
2 INITIPADDR 192.59.13.229
2 INITIISADDR 192.168.85.2
2 INITIISPORT 80
2 INITBROWSERS 440
2 INITMYWID 45,88
//
//  Configuration Driver 3
//
3 INITIPADDR 192.59.13.230
3 INITIISADDR 192.168.87.3
3 INITIISPORT 80
3 INITBROWSERS 440
3 INITMYWID 89,132
//
//  Configuration Driver 4
//
4 INITIPADDR 192.59.13.231
4 INITIISADDR 192.168.89.4
4 INITIISPORT 80
4 INITBROWSERS 440
4 INITMYWID 133,176
//
//  Configuration Driver 5
//
5 INITIPADDR 192.59.13.223
5 INITIISADDR 192.59.32.5
5 INITIISPORT 80
5 INITBROWSERS 440
5 INITMYWID 177,220
//
//  Configuration Driver 6
//
6 INITIPADDR 192.59.13.224
6 INITIISADDR 192.59.34.6
6 INITIISPORT 80
6 INITBROWSERS 440
6 INITMYWID 221,264
//
//  Configuration Driver 7
//
7 INITIPADDR 192.59.13.224
7 INITIISADDR 192.59.35.6
7 INITIISPORT 80
7 INITBROWSERS 440
7 INITMYWID 265,308
//
//  Configuration Driver 8
//
8 INITIPADDR 192.59.13.223
8 INITIISADDR 192.59.33.5
8 INITIISPORT 80
8 INITBROWSERS 440
8 INITMYWID 309,352
//
```

```
//  Configuration Driver 9
//
9 INITIPADDR 192.59.13.231
9 INITIISADDR 192.168.90.4
9 INITIISPORT 80
9 INITBROWSERS 440
9 INITMYWID 353,396
//
//  Configuration Driver 10
//
10 INITIPADDR 192.59.13.230
10 INITIISADDR 192.168.88.3
10 INITIISPORT 80
10 INITBROWSERS 440
10 INITMYWID 397,440
//
//  Configuration Driver 11
//
11 INITIPADDR 192.59.13.229
11 INITIISADDR 192.168.86.2
11 INITIISPORT 80
11 INITBROWSERS 440
11 INITMYWID 441,484
//
//  Configuration Driver 12
//
12 INITIPADDR 192.59.13.228
12 INITIISADDR 192.168.84.1
12 INITIISPORT 80
12 INITBROWSERS 440
12 INITMYWID 485,528
//
//  Configuration Driver 13
//
13 INITIPADDR 192.59.13.228
13 INITIISADDR 192.168.83.1
13 INITIISPORT 80
13 INITBROWSERS 440
13 INITMYWID 529,572
//
//  Configuration Driver 14
//
14 INITIPADDR 192.59.13.229
14 INITIISADDR 192.168.85.2
14 INITIISPORT 80
14 INITBROWSERS 440
14 INITMYWID 573,616
//
//  Configuration Driver 15
//
15 INITIPADDR 192.59.13.230
15 INITIISADDR 192.168.87.3
15 INITIISPORT 80
15 INITBROWSERS 440
15 INITMYWID 617,660
//
//  Configuration Driver 16
//
16 INITIPADDR 192.59.13.231
16 INITIISADDR 192.168.89.4
16 INITIISPORT 80
16 INITBROWSERS 450
16 INITMYWID 661,705
//
//  Configuration Driver 17
//
17 INITIPADDR 192.59.13.223
17 INITIISADDR 192.59.32.5
17 INITIISPORT 80
17 INITBROWSERS 450
17 INITMYWID 706,750
//
//  Configuration Driver 18
//
18 INITIPADDR 192.59.13.224
18 INITIISADDR 192.59.34.6
18 INITIISPORT 80
18 INITBROWSERS 450
18 INITMYWID 751,795
//
//  Configuration Driver 19
//
19 INITIPADDR 192.59.13.224
19 INITIISADDR 192.59.35.6
19 INITIISPORT 80
19 INITBROWSERS 450
19 INITMYWID 796,840
//
//  Configuration Driver 20
//
20 INITIPADDR 192.59.13.223
20 INITIISADDR 192.59.33.5
20 INITIISPORT 80
20 INITBROWSERS 450
20 INITMYWID 841,885
//
//  Configuration Driver 21
//
21 INITIPADDR 192.59.13.231
21 INITIISADDR 192.168.90.4
21 INITIISPORT 80
21 INITBROWSERS 450
21 INITMYWID 886,930
//
//  Configuration Driver 22
//
22 INITIPADDR 192.59.13.230
22 INITIISADDR 192.168.88.3
22 INITIISPORT 80
22 INITBROWSERS 450
22 INITMYWID 931,975
//
//  Configuration Driver 23
//
23 INITIPADDR 192.59.13.229
23 INITIISADDR 192.168.86.2
23 INITIISPORT 80
23 INITBROWSERS 450
```

```
23 INITMYWID 976,1020
//
//  Configuration Driver 24
//
24 INITIPADDR 192.59.13.228
24 INITIISADDR 192.168.84.1
24 INITIISPORT 80
24 INITBROWSERS 450
24 INITMYWID 1021,1065
//
```

Driver Environment

Note: this configuration file is repeated on each of the other 23 drivers with the exception of `WEBDRIVERNO`, which is replaced by 2 thru 24.

```
set WEBDRIVERNO=1
set WEBADMBASEPORT=4300
set WEBDIAGLEVEL=2
set WEBEVENTLOG=1
set WEBEVENTHOST=
set WEBLOGLEVEL=1
set WEBSINGLETRAN=0
set WEBTPCCAUDIT=0
set WEBRTFUDGETM=110
set WEBNEWORDERPROB=4476
set WEBPAYMENTPROB=4309
set WEBORDERSTATUSPROB=405
set WEBDELIVERYPROB=405
set WEBSTOCKLEVELPROB=405

webdriver.exe
exit
```

Appendix E - Disk Storage

TPC-C 180-Day Disk Space Requirements

Warehouses	**1110**	tpmC	**13089.30**		
Table	**Initial Rows**	**Data KB**	**Index KB**	**Extra 5% KB**	**Total With 5% KB**
Warehouse	1,110	2,220	12	112	2,344
District	11,100	22,200	94	1,115	23,409
Customer	33,300,000	22,204,440	1,723,374	1,196,391	25,124,205
History (D)	33,300,000	1,665,002	0		1,665,002
Order (D)	33,300,000	865,800	5,222		871,022
New-Order	9,990,000	111,000	674	5,584	117,258
Order-Line (D)	332,999,230	18,510,758	120,992		18,631,750
Item	100,000	9,100	46	457	9,603
Stock	111,000,000	37,007,400	204,468	1,860,593	39,072,461
Totals KB		80,397,920	2,054,882	3,064,251	85,517,053

Dbspaces	**Count**	**Size MB**	**MB Allocated**	**MB Loaded +5%**	**MB for 8 Hours**
master, model, tempdb & msdb		43	43	43	43
tpc_misc	4	655	3,500	2,626	3,103
	1	880			
tpc_cs	5	9,952	63,000	62,692	62,692
	1	13,240			
tpc_ol	4	4,549	22,500	18,195	21,662
	1	4,304			
Total Allocated MB			**89,043**	**83,556**	**87,500**

Dynamic Space MB	20,548	Sum of data for orders, order_line & history
Static Space	62,964	Sum of data+index+5% - Dynamic Space
Free Space	5,530	Total allocated space - (Dynamic & Static Spaces)
Daily Growth	3,877	(Dynamic Space / (W * 62.5)) * tpmC
Daily Spread	0	Free space - 1.5 * Daily growth (zero if negative)
	0	SQL Server can be configured to eliminate Daily Spread
180 Day Space MB	760,817	Static Space + 180 * (Daily Growth + Daily Spread)
180 Day Space GB	**742.99**	
8 hr log GB	**33.24**	(need double for mirroring)
Disk Capacity MB	4149	**4.0518 GB** Capacity of 4GB disks
	8682	**8.4785 GB** Capacity of 9GB disks

Space Usage	**GB Needed**	**Disks Priced**	**GB Priced**	
180-day space DB	742.99 GB	81	328.19 GB	4GB drives
		40	339.14 GB	9GB drives
	extra disks	9	76.31 GB	9GB drives
Total DB		130	743.64 GB	
8-hr log+mirror	66.48 GB	8	67.84 GB	9GB drives
OS, SQL Server	4.00 GB	1	4.05 GB	4GB drives
Total space	**813.47 GB**	**139**	**815.53 GB**	

TPC-C 180-Day Dynamic Table Growth Rates

Table	Initial (KB)	Final (KB)	Change(KB)	KB per New-Order	8-Hr
History	1,665,002	1,886,456	221,454	0.0486	
Orders	871,022	1,003,960	132,938	0.0292	
misc_seg	2,536,024	2,890,416	354,392	0.0778	477.5099
Order_line	18,631,750	21,204,964	2,573,214		
ol_seg	18,631,750	21,204,964	2,573,214	0.5651	3467.1637
Syslogs	6	25,262,838	25,262,832		
logsegment	6	25,262,838	25,262,832	5.5478	33.2415
SUM(d_next_o_id)	33,311,100	37,864,746	4,553,646		
New_order	111,674	135,654	23,980	0.0053	

Appendix F - Third-Party Price Quotations

NETLUX

14180 Live Oak Ave., Unit E
Baldwin Park, Ca. 91760

1-800-789-1780
Phone #818-851-9737
Fax #818-851-9837

October 14, 1997

UNISYS Corp.
Rick Freeman
25725 Jeronimo Rd.
Mission Viejo, Ca 92691
714-380-5344

Quotation

Quantity	Part No.	Description	Unit Price	Total
3	NX-H8TX	8-port 100Mbps FAST Ethernet Hub	$299.00	$ 897.00

Terms and Conditions:
FOB Origin
5 Year Warranty
Prices good for 60 Days

Sincerely,
Martin Parry
NETLUX

NETLUX

14180 Live Oak Ave., Unit E
Baldwin Park, Ca. 91760

1-800-789-1780
Phone #818-851-9737
Fax #818-851-9837

October 14, 1997

UNISYS Corp.
Rick Freeman
25725 Jeronimo Rd.
Mission Viejo, Ca 92691
714-380-5344

Quotation

Quantity	Part No.	Description	Unit Price	Total
736	NX-H16EZ	16-port Desktop Ethernet Hub	$83.00	$61,088.00

Terms and Conditions:
FOB Origin
5 Year Warranty
Prices good for 60 Days

Sincerely,
Martin Parry
NETLUX

BEA ENTERPRISE MIDDLEWARE SOLUTIONS

May 16, 1997

Jerrold Buggert
Director, Unisys System Analysis' Modeling, and Measurement
Unisys
25725 Jeronimo Road
Mission Viejo, CA. 92691

Subject: Pricing for Tuxedo 6.3

Dear Mr. Buggert;

This letter is to confirm our policy on Tuxedo versions' general availability and prices. At present we are in the process of removing Tuxedo 4.2.2 (from BEA, ITI, IMC or Novell) from general availabilty. It is replaced by Tuxedo 6.x from BEA Systems, Inc.

For purposes of TPC activity we recommend the use of Tuxedo 6.3 available through our Core Functionality Services (CFS) program. This is a replacement for version 4.2.2 or earllier. This is a genrally available product and should be denoted as Tuxedo 6.3 CFS in all publications. It is priced as shown below.

BEA Tuxedo Core Functionaty Services Program License Fees Per Server

Unlimited User License fees per server	**Number of Users**	**Dollar Amount**	**Maintenance (5 x 8) per year**	**Maintenance (7 x 24) per year**
Tier 1 -- PC Servers with 1 or 2 CPUs, RISC Uniprocessor Workstations	Unlimited	$3,000.00	$450.00	$660.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange and UNIX Systems	Unlimited	$12,000.00	$1,800.00	$2,640.00
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	$30,000.00	$4,500.00	$6,600.00
Tier 4 - Large (8 - 32 CPUs) and Mainframe Systems	Unlimited	$100,000.00	$15,000.00	$22,000.00
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$37,500.00	$55,000.00

Sincerely,

Lew Brentano
Director, Product Management

Western Micro Technology
(800)937-8446

12/18/97

Quoted to: Jill Christman/Unisys for TPC.org
Prepared by: Bill Scott

Qty.	Description	Style	Unit Price	Extended Price
	Server Hardware			
1	SYS: Aquanta HS/6, 0 Proc, 0MB Mem	HS6000122-BAS	$9,985	$9,985
1	PROC:3x200MHz PPro/1MB + CPU Riser card	HTC6200-1MB	$13,630	$13,630
1	PROC:3x200MHz PentiumPro/1MB Cache	HTX6200-1MB	$13,306	$13,306
1	MEM:ECC Memory Board, 0MB Mem	MEM641-DIM	$395	$395
16	MEM:256 MB Memory Upgrade	DIM572-256	$2,048	$32,768
7	CTRL:RAID Tri-SCSI-2 Ultra PCI	RAD3162-PCI	$2,074	$14,518
1	CTRL: VGA, 16-bit ISA	VID11-ISA	$106	$106
1	ETHERNET: 100Mbit/sec, PCI 32-bit	SF1001-FET	$290	$290
1	CDROM: 16 Speed	CDR1200-SI	$178	$178
1	ACC: 6 SCA Drive Cage	CAG61-ADV	$425	$425
91	DISK: 4GB Drive + 10% spares	HDS4000-WC7	$743	$67,613
54	DISK: 9GB Drive + 10% spares	HDS9000-WC7	$1,495	$80,730
10	DISK: 9GB Drive + 10% spares	HDS9000-H10	$1,602	$16,020
1	MONITOR:14-inch Color	EVG142-COL	$219	$219
1	KEYBD: 104 Key Spacesaver	PCK104-SKB	$34	$34
1	MOUSE: 2 Button PS2	PWM1-PS2	$25	$25
2	PWR: 2200VA UPS, 4U	UPD22001-SXR	$2,380	$4,760
3	CAB: Rack Cabinet, w/ fill pnls, 36U	CAB361-SXR	$1,530	$4,590
2	CAB: Link kit for 36U cabinets	LNK361-SXR	$255	$510
3	CAB: Bezel kit 36U	BEZ3611-CAB	$170	$510
3	CAB: Stabilizer kit 0U	WGT39581-SXR	$120	$360
1	PNL: L&R side panels 36U	PAN3621-SXR	$213	$213
	System Total			**$261,185**
	Client Hardware			
6	SYS: Aquanta DM/6 II, 0 Proc, 0MB Mem	CMT60072-ZFA	$896	$5,376
6	PROC:1x200MHz PentiumPro/256KB Cache	PRC6200-256	$726	$4,356
24	MEM: 64 MB Memory Upgrade	MTP6-64M	$634	$15,216
6	CTRL: VGA, 64-bit with 2MB	PCV102-PCI	$123	$738
6	DISK: 1.6GB IDE 3.5 Internal	HDI1600-5	$219	$1,314
6	CDROM: Twelve Speed IDE	CDR1200-AI	$126	$756
6	ETHERNET: 100Mbit/sec, PCI 32-bit	ETH101007-PCI	$120	$720
12	ETHERNET: 10Mbit/sec, PCI 32-bit	ETH101-PCI	$130	$1,560
6	MONITOR:14-inch Color	EVG142-COL	$219	$1,314
6	KEYBD: 104 Key Spacesaver	PCK104-SKB	$34	$204
6	MOUSE: 2 Button PS2	PWM1-PS2	$25	$150
	Systems Total			**$31,704**

Quote valid for 75 days.

ALR
Advanced Logic Research, Inc.
9401 Jeronimo Road • Irvine • California 92718
(800) 444-4257 • (714) 581-9240 Fax

October 13, 1997

Rick Freeman
Unisys Corporation
25725 Jeronimo Road
Mission Viejo, CA 92691

Dear Rick,

Following please find the prices for the products/services you have requested.

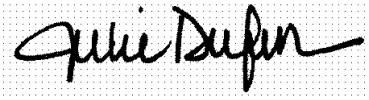
Quick Hot Swap Storage Drawer II (7u) #11910190 $1,795 SUP

5 Year Warranty/Factory Depot Service (ALR-Irvine, CA) for the above mentioned drawer (11910190). This price is based on a one-time purchase of 10-16 drawers for each service, and is based upon a 7-day repair or replenishment turn around time.

Price per drawer: $431.00 each.

If you have any questions please contact me immediately at (800) 444-4257 x 2260.

Sincerely,



Julie Dufur
Account Manager/OEM Sales

cc: Glenn Weeks, Unisys Corporation
Bob Dunford, Advanced Logic Research
Ben Marchak, Advanced Logic Research

Document: rf101397.DOC

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 206 882 8080
Telex 160520
Fax 206 936 7329

Microsoft

October 17, 1997

Mr. Jerrold Buggert
Director, Systems Analysis, Modeling, Measurement
Unisys Corporation
25725 Jeronimo Road
Mission Viejo, CA 92691

via FAX # 714-380-5468

Dear Jerry,

Here is the information you requested regarding pricing of certain Microsoft products:

Microsoft SQL Server 6.5, Enterprise Edition, unlimited user licence	$28999
Microsoft Windows NT Server 4.0, Enterprise Edition, incl 25 CALs	$3999
Windows NT Server 4.0 software, incl 5 CALs	$809
Microsoft SQL Workstation (includes programmers toolkit)	$499
Visual C++ 32-bit edition (subscription)	$499
5-yr maintenance for above software @ $2095/yr	$10475

This quote is valid for the next 60 days. Please let me know if I can be of any further assistance.

Best regards,

Sid Arora

Sid Arora
Product Manager, Microsoft SQL Server
Personal and Business Systems Group

Microsoft Corporation is an equal opportunity employer.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 206 882 8080
Telex 160520
Fax 206 936 7329

Microsoft

October 28, 1997

Mr. Jerrold Buggert
Director, Systems Analysis, Modeling, Measurement
Unisys Corporation
25725 Jeronimo Road
Mission Viejo, CA 92691

via FAX # 714-380-5539

Dear Jerry,

Microsoft has received your request for permission to disclose the results of TPC-C benchmark tests conducted by Unisys with Microsoft SQL Server, Enterprise Edition 6.5 on the following system:

Unisys Aquanta HS/6 Server, 6-processors, Pentium Pro, 200 MHz, 1MB cache
Test Results: 13089 tpmC @ $39/tpmC approximately

Microsoft hereby grants Unisys permission to disclose these results to third parties and acknowledges that Unisys has formally requested permission to do so in accordance with the license agreement for Microsoft SQL Server Enterprise Edition software.

Best regards,

Sid Arora
Product Manager, Microsoft SQL Server
Personal and Business Systems Group

Microsoft Corporation is an equal opportunity employer.