Hewlett-Packard Company
Network Server Division

HP NetServer LXr Pro8
Using Microsoft SQL Server 6.5 on Microsoft NT 4.0 Enterprise Edition

TPC Benchmark® C Full Disclosure Report

First Edition
March 27, 1998

First Edition - March 27, 1998 First Printing.

Hewlett-Packard Company 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. Hewlett-Packard Company 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, Hewlett-Packard Company provides no warranty of the pricing information in this document.

Benchmark results are highly dependent upon workload, specific application requirements, and system 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. Results obtained in other operating environments may vary significantly. Hewlett-Packard Company does 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.

Printed in U.S.A., March 27, 1998

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the NetServer LXr Pro8 in a client/server configuration, using Microsoft SQLServer 6.5™ and the TUXEDO transaction monitor. The operating system used for the benchmark was Microsoft NT Server 4.0 Enterprise Edition. The application was written in C and compiled using Microsoft Visual C++.

TPC Benchmark® C 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.

Standard and Executive Summary Statements

Page ***iv*** contains the standard system summary and pages ***v***- contain the executive summary of the benchmark results for the HP NetServer LXr Pro8 system.

Auditor

The benchmark configuration, environment and methodology used to produce and validate the test results, and 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 specifications.

Standard System Summary

Company Name	System Name	Database Software	Operating System Software
Hewlett-Packard Co.	Hewlett-Packard NetServer LXr Pro8 SMP (8-way)	Microsoft SQL Server v6.5 Enterprise Edition	Microsoft NT Server 4.0 Enterprise Edition
Availability Date: January, 1998			

Total System Cost	TPC-C® Throughput	Price/Performance
Hardware Software 5-year maintenance	Sustained maximum throughput of system running TPC-C® expressed in transactions per minute	Total system cost/tpmC® ($547,374/16257.20)
$547,374	**16257.20 tpmC®**	**$33.67 per tpmC®**

Additional copies of this Full Disclosure Report can be obtained from either the Transaction Processing Performance Council or Hewlett-Packard Company at the following address:

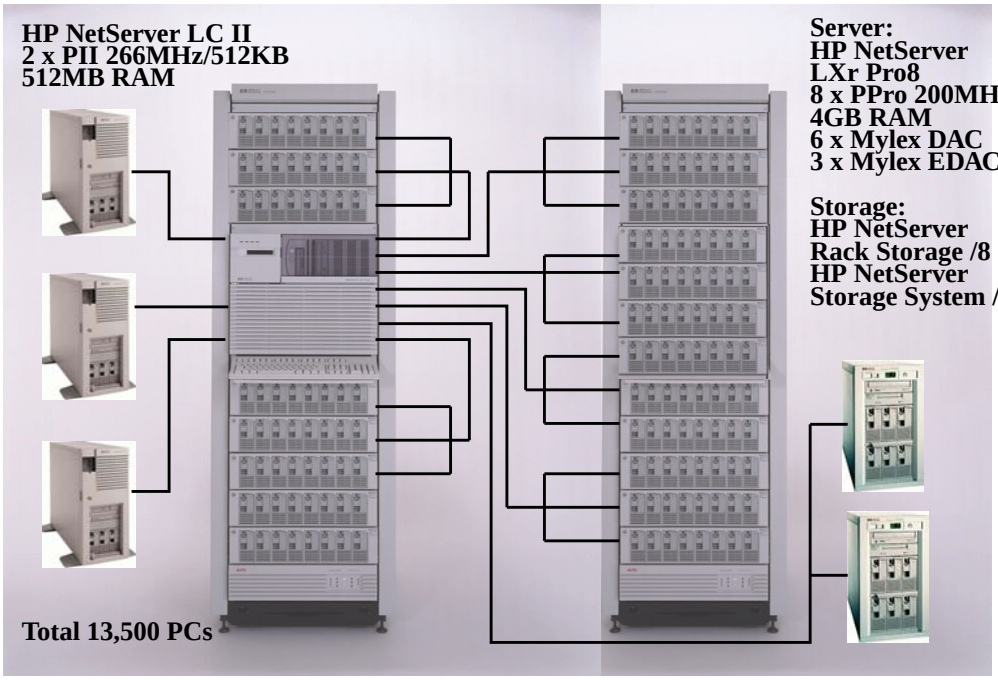
Transaction Processing Performance Council (TPC)
c/o Shanley Public Relations
777 North First Street, Suite 600
San Jose, CA 95112, USA
Phone: (408) 295-8894, (408) 295-9768 fax

or

Hewlett-Packard Company/NetWork Server Division
10450 Ridgeview Court
Cupertino, CA 95015-4050 USA

Attention: Adrianna Ma, Bldg. 49EL MS FS

Hewlett-Packard Co.	NetServer LXr Pro8 Client/Server with 3 NetServer LC II Front-Ends			TPC-C® Rev 3.3 Report Date: March 27, 1998
Total System Cost	TPC-C® Throughput		Price/Performance	Availability Date
$547,374	16257.20 tpmC®		$33.67 / tpmC®	January, 1998
Processors	Database Manager	Operating System	Other Software	Number of Users
8 Intel Pentium Pro 200MHz/1MB	Microsoft SQL Server v. 6.5 Enterprise Edition	Microsoft NT Server 4.0 Enterprise Edition	Microsoft C++ Compiler TUXEDO Transaction Monitor	13,500



System Components	*Server (LXr Pro 8)*		*Each Client (LC II)*	
	Qty	*Type*	*Qty*	*Type*
Processors	8	200 MHz Intel Pentium Pro	2	266 MHz Intel Pentium II
Cache memory	*each*	1MB cache	*each*	512 KB cache
Memory	4	GB	512	MB
Disk Array Controllers	9	Mylex Disk Array Controller	1	HP SCSI-2 Controller
Disk Drives	107 48	HP Hot-Swap 9Gbyte SCSI HP Hot-Swap 4Gbyte SCSI	1	4 Gbyte Disk
Total Storage (TB)	1.16	TB		
Tape Drives	1	DAT Storage System		
Terminals	1	Console terminal	1	Console terminal

HEWLETT PACKARD
Network Server Division

HP NetServer LXr Pro8
8 way Pentium Pro, 4GB RAM
Microsoft Windows NT 4.0 Enterprise,
SQL Server 6.5 Enterprise
Client/Server

TPC-C Rev. 3.3

Report Date:
March 27, 1998

	Product		Price	Purchase Price Unit	Qty	Purchase Price Extended	Support Price Unit	Qty	Support Price 5 Year
Server Hardware	Number	Brand	Key	Price		Price	$/Month		Support
HP NetServer LXr Pro8 6/200 Model 1	D5028A	HP	1	$28,260	1	$28,260			
3 year, on-site, next day warranty	incl								
2 200MHz Pentium Pro processor each w/1MB L2 cache	incl								
256 MB memory	incl								
Integrated dual PCI F/W SCSI-2 cntrlrs	incl								
1024x768 16 color video 512KB video memory	incl								
CD-ROM drive, 3.5-inch, 1.44MB floppy drive	incl								
1 parallel, 2 serial ports, keyboard & mouse	incl								
3 hot-swap power modules & N+1 redundant fans	incl								
HP NetServer Navigator, HP NetServer Assistant	incl								
HP Open View for Windows	incl								
NetServer LX 6/200 Dual Processor card with:	----	HP							
2 Pentium Pro 200MHz chips each with 1MB L2 cache	D5030A		1	$11,558	3	$34,674			
HP 10/100 PCI Network Adaptor	J3171A	HP	1	$72	1	$72			
14" VGA Monitor	D2821S	HP	1	$205	1	$205			
HP NetServer mini-DIN keyboard and mouse	D4950B/C3751B	HP	1	$126	1	$126			
APC Smart UPS 3000NS 208V 3000VA	588293	HP	1	1,740	1	$1,740			
Power Distribution Unit	E7675A	HP	1	$169	4	$676			
Power Cords	E7802A	HP	1	$46	4	$184			
2.0 Meter Rack w/side panels and rear door	J1487B	HP	1	$1,470	2	$2,940			
HP NetServer Rack installation Kit	D4984B	HP	1	$54	1	$54			
5 Yr Support (upgds wrnty to same day, 4 hr)	H5526A+O2A	HP	1, 2						$6,586
5 Yr Support - 14" VGA Monitor		HP							$168
SPU Total	**$68,931**								
256 MByte ECC DIMM Memory Module	D5026A	HP	1	$2,310	15	$34,650			
Memory Total	**$34,650**								
Storage									
HP 9GB SCSI-2 hot swap disk + 10% spares	D4289A	HP	1	$1,229	106	$130,293			
HP 9GB SCSI-2 hot swap disk + 10% spares	D6019A	HP	1	$1,675	12	$20,100			
HP 4.2 GB SCSI-2 hot swap disk + 10% spares	D3583C	HP	1	$721	53	$38,201			
HP 9 GB SCSI-2 disk drive (common tray)	D4911A	HP	1	$1,260	1	$1,260			
HP Storage System/6 + 10% spares	D3604B	HP	1	$1,015	5	$5,075			
HP Rack Storage/8 + 10% spares	D4902A	HP	1	$2,100	22	$46,200			
Mylex DACPJ-3-8E F/W SCSI Ctlr 3 chnl+ 10% spares	Z75090	Myl	1	$1,555	8	$12,440			$564
64 MB Upgrade to DACPJ-3	ADC64-DAC	Myl	1	$159	6	$954			
Mylex EDAC (32MB) + 10% spares	CSxi-5-6	Myl	1	$3,295	5	$16,475			
NetRAID 3rd channel SCSI to RS/8	D4983A	HP	1	$86	9	$774			
External SCSI cable RS/8 + 10% spare	D3637C	HP	1	$86	14	$1,204			
SCSI cable 2.5m	D3636C	HP	1	$86	9	$774			
SCSI cable 0.9m	C2911A	HP	1	$124	4	$496			
HP SureStore Tape 6000i 8Gb Internal DAT	C1528F	HP	1	$905	1	$905			
Storage total	**$34,022**								
Server Software									
Microsoft Windows NT Server 4.0 Enterprise Edition	779-00001	MS	1	$3,345	1	$3,345			
Microsoft SQL Server 6.5 Enterprise Edition	810-00007	MS	1	$24,410	1	$24,410			$10,475
Server software Total	**$27,755**								
Connectivity									
Compex Microhub 6port 100BaseT + 10% spares	MX1205		1	$189	3	$567			
Netlux Ethernet Hub, 16-port 10Base-T +10%spares	NX-H16EZ		1	$75	990	$74,250			
Connectivity Total	**$74,817**								
Client Hardware									
HP NetServer LC II PII/266 Model 4	D5014A	HP	1	$2,375	3	$7,125			
HP Pentium II 266 MHz CPU Upgrade	D4995A	HP	1	$924	3	$2,772			
128MB DIMMS 60ns with ECC	D4297A	HP	1	$688	12	$8,256			
HP NetServer 10/100TX PCI LAN Adapter	D5013A	HP	1	$83	15	$1,245			
14" VGA Monitor	D2821S	HP	1	$205	3	$615			
5 Yr Support - 14" VGA Monitor		HP	2						$504
5 Yr Support - LC II SPU and all internal components	H5519A+02A	HP	1,2						$5,664
Client Total	**$20,013**								
Client Software									
Windows NT 4.0 Server	ZS750001	MS	1	$475	3	$1,425			
Tuxedo Run Time	--	BEA	3	$3,000	3	$9,000			$6,750
Tuxedo Developers Kit	--	BEA	3	$2,495	1	$2,495			$1,871
SQL Svr Programmer's toolkit	ZS750002	MS	1	$129	1	$129			
Microsoft C++ v. 4.0	ZS750003	MS	1	$425	1	$425			
Client s/w Total	**$13,474**								
Net purchase & support prices						**$514,791**			**$32,583**

Notes:

1. Support pack H5520A upgrades standard NetServer 3 year warranty to same day, 4-hr response, which includes monitor, disks and SPU internal components, e.g. (memory,NIC ,DAC, DAT, etc.) for 3 years. NetServer support for years 4-5 via HP; contract support (order opt. 02A for 24 months), includes all HP components in SPU. $7 per month for monitors.
2. PriceKey: 1=Software house, 2=HP Corporate Price List, 3=BEA Systems

5 Yr C-O-O	547,374
tpmC	16257.20
$/tpmC	$33.67

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 assumptions 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 benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc org. Thank you.

Numerical Quantities Summary for NetServer LXr Pro8

MQTH, Computed Maximum Qualified Throughput 16257.20 tpmC®

Response Times (in seconds)	90th%-ile	Maximum	Average
New-Order	1.36	4.98	0.82
Payment	1.11	4.67	0.63
Order-Status	2.79	6.49	1.74
Delivery (interactive portion)	0.70	2.38	0.37
Delivery (deferred portion)	1.30	3.80	0.84
Stock-Level	5.21	8.09	3.45
Menu	0.60	3.59	0.26

Transaction Mix, in percent of total transactions

New-Order	44.83%
Payment	43.06%
Order-Status	4.05%
Delivery	4.03%
Stock-Level	4.03%

Keying/Think Times (in seconds)	Keying Time			Think Time		
	Min.	Avg.	Max.	Min.	Avg.	Max.
New-Order	18.01	18.02	18.17	0.01	12.13	168.66
Payment	3.01	3.02	3.17	0.01	12.17	174.52
Order-Status	2.01	2.02	2.16	0.01	10.14	107.33
Delivery (interactive)	2.01	2.02	2.16	0.01	5.14	59.96
Stock-Level	2.01	2.02	2.16	0.01	5.18	65.49

Numerical Quantities Summary for NetServer LXr Pro8, continued

Test Duration

Ramp-up time		43.60 minutes
Measurement interval		30 minutes
Transactions during measurement interval		1087974
Ramp down time		11.40 minutes

Checkpointing

Number of checkpoints in measurement interval		1
Checkpoint interval		30 minutes

Reproducibility Run

	16129.77C	-0.78%

Preface

This is the full disclosure report for a benchmark test of the NetServer LXr Pro8 using Microsoft SQLServer 6.5 Enterprise Edition. It meets the requirements of the TPC Benchmark® C Standard Specification, Revision 3.3 dated April 8, 1997.

TPC Benchmark® C was developed by the **T**ransaction Processing **P**erformance **C**ouncil (TPC). It is the intent of this group to develop a suite of benchmarks to measure the performance of computer systems executing a wide range of applications. Hewlett-Packard Company and Microsoft, Inc. are active participants in the TPC.

TPC Benchmark® C Overview

*TPC Benchmark® C is an **O**n **L**ine **T**ransaction **P**rocessing (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 of 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 applications, 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 other environments are not recommended.

Hewlett-Packard Company does not warrant or represent that a user can or will achieve performance similar to the benchmark results contained in this report. No warranty of system performance or price/performance is expressed or implied by this report.

System Overview

The hardware configuration used in this TPC-C test was based on the Hewlett-Packard NetServer LXr Pro8. The full configuration was built by adding additional memory, additional disk adapters and drives, and a network adaptor. The operating system used was Microsoft's NT 4.0 Enterprise Edition and the database was Microsoft's SQL 6.5 Enterprise Edition.

The architecture of the NetServer LXr Pro8 was designed for the Intel Pentium Pro chip and associated chipset. The NetServer LXr Pro8 used in this test was powered by eight 200 MHz Intel Pentium Pro(R) processor chips, each with 1Mbyte of SRAM 2nd level cache. In the NetServer LXr Pro8, four separate dual processor cards are used, each containing two Pentium Pro chips. Within the cards there is an interface between the two chips called the P6 bus. Both of these processor cards plug directly into a motherboard. The interface between the motherboard and the processor cards is an extension of the same P6 bus.

This configuration used 4 Gbytes of HP RAM. This was achieved by adding 28 128 Mbyte DIMMs to the 4 128 Mbyte DIMM that came with the memory board. This RAM was attached to the system bus via a controller.

This configuration also used SCSI-2 Fast/Wide PCI Disk controllers embedded in the motherboard, six Mylex 3-channel PCI Disk Array Controllers (DACs), and 3 Mylex External DACs (EDACs). The DACs are plugged into PCI slots on the motherboard, which are divided between two separate 33MHz PCI I/O buses. Both PCI busses are attached directly to the P6 bus through separate PCI bridges so that PCI bus masters have direct access to memory.

One HP 9Gbyte SCSI-2 (common tray) Fast hard disk and one CD-ROM drive were attached to one of the embedded PCI SCSI controllers. This disk drive was used for the Operating System (NT v4.0 Enterprise Edition), all executables and libraries, the master database, and swap space.

In the measured configuration, five pairs of the 9 Gbyte HP SCSI-2 Hot Swap hard disks attached to three Mylex EDACs, which are connected to the embedded PCI SCSI controllers were used exclusively for the database log. Forty-eight HP 4Gbyte Hot Swap drives and 96 9Gbyte Hot Swap disk drives were equally distributed across the 6 3-Channel Mylex PCI Disk Array Controllers (DACs). Eight Hot Swap disks were assigned per DAC SCSI channel. Each channel was striped and the channels spanned using the Mylex Utility. Controller write-back caching and read ahead were specifically disabled for the PCI DACs.

At the operating system level, NT's disk administrator shows 8 disk drives - the 9Gbyte SCSI-2 boot drive, one 43,389 Mbyte disk drive containing the hardware mirrored sets of 9Gbyte Hot Swap drives used for the log, four 208,272 Mbyte disk drives externalized by the Mylex DACs containing the 9Gbyte disks, and two 97,608 Mbyte disks externalized by the Mylex DACs with 4Gbyte disk drives.

The six logical disk drives were used to hold the TPC database. This was done to maximize performance. Protection against data loss from a failed drive was achieved by normal database level recovery and from the NT mirrored log drives.

This configuration also used one HP J3171A NetServer PCI network adaptor card, attached to the NetServer LXr Pro8 motherboard via the PCI bus. This network adaptor supplied a 10BaseT network interface to the three NetServer clients. Each of the clients had 512 Mbytes of RAM, one 4 Gbyte SCSI hard disk, one HP D5013 NetServer PCI network adaptor card, and was running NT 4.0. HP VGA displays were used on the NetServer LXr Pro8 and on each of the three clients.

Section 1.0 – General Items

1.1 Test Sponsor

A statement identifying the sponsor of the Benchmark and any other companies who have participated.

The Network Server Division of the Hewlett-Packard Company was the test sponsor of this TPC Benchmark C.

1.2 Application Code and Definition Statements

The application program must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input/output functions.

The Section 3.0 entitled Clause 3 Related Items contains a brief discussion of the database design and loading. The database definiton statements, distribution across disk drives, loading scripts, and tables are provided in Appendix A- Database Generation

The program that implements the TPC Benchmark C translation and collects appropriate transaction statistics is referred to as the Remote Terminal Emulator (RTE) or Driver program. The Driver program is discussed in Section 7.0. The source code for this driver program is provided in Apprendix B - Source Code.

1.3 Parameter Settings

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

- *Database options*
- *Recover/commit options*
- *Consistency/locking options*
- *System parameter, application parameters, and configuration parameters.*

This requirement can be satisfied by providing a full listing of all parameters and options.

Appendix C contains all the database and operating system parameters used in this benchmark. Appendix D contains all the hardware configuration details.

1.4 Configuration Diagrams

Diagrams of both the measured priced system must be provided, accompanied by a description of the differences.

Figure 1-1 and 1-2 show the measured and priced client/server configurations. The SUT in the measured system is idential to the priced one.

FIGURE 1-1: NetServer LX Pro - Measured Configuration

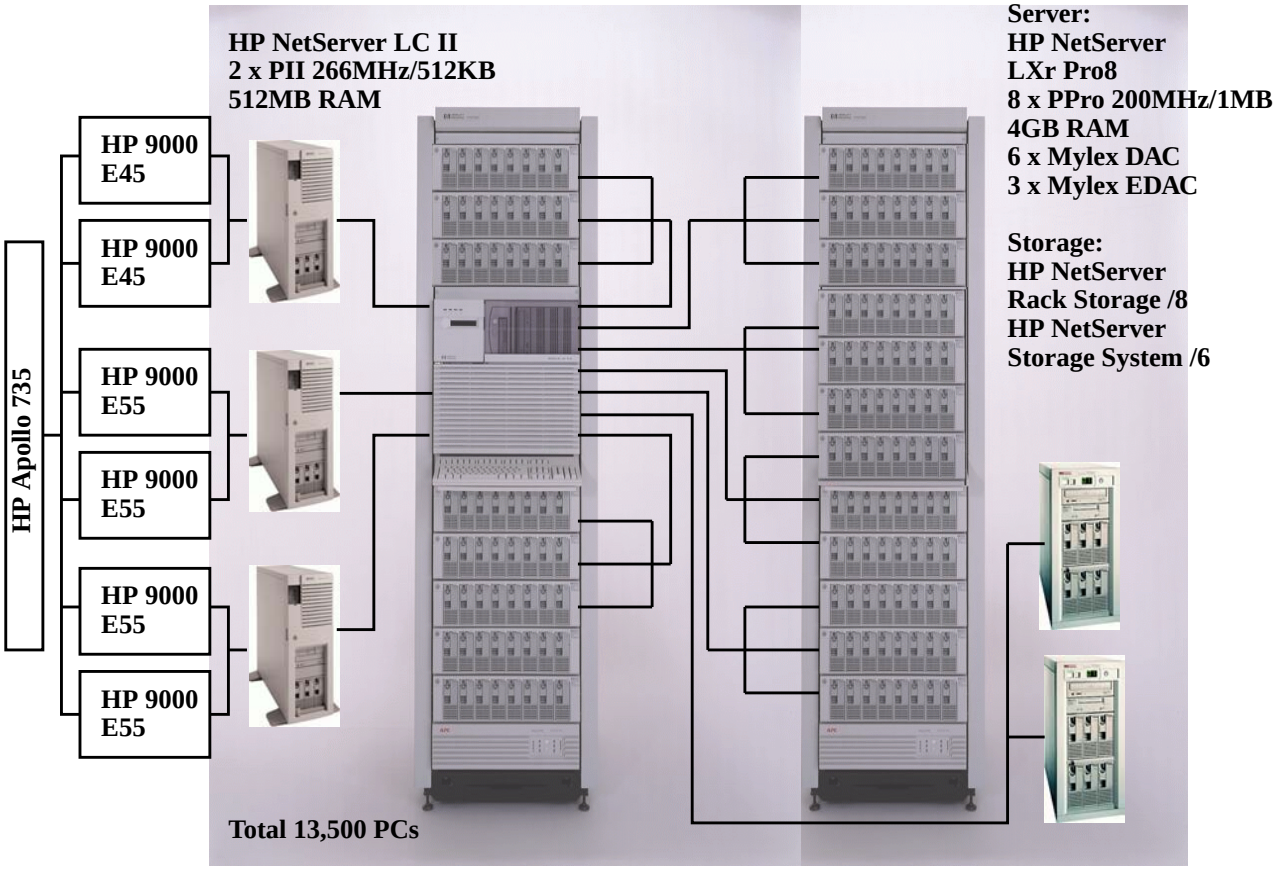
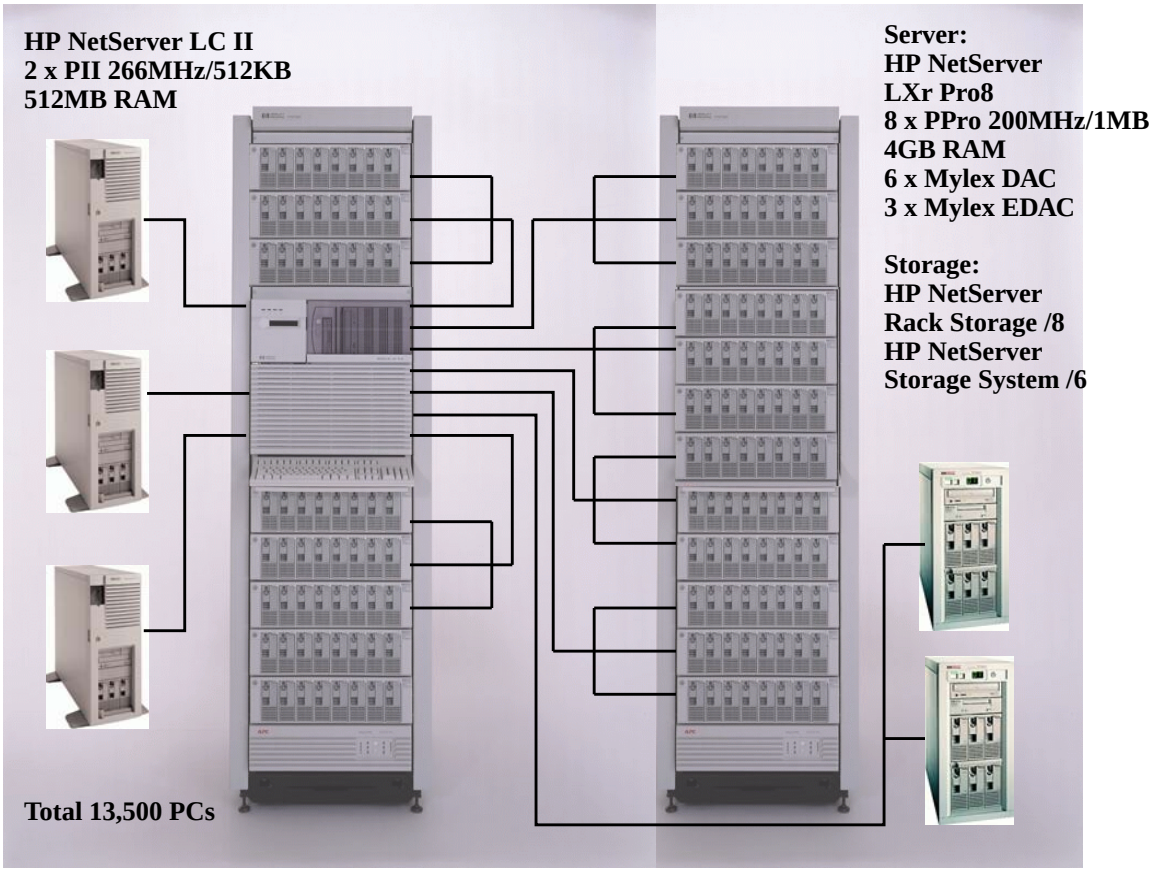


FIGURE 1-2: NetServer LX Pro - Priced Configuration



Section 2.0 – Clause 1 Related Items

2.1 Table Definitions

A listing must be provided for all table definitions statements and all other statements used to set up the database.

Appendix B contains the code used to define and load the database tables.

2.2 Physical Organization of the Database

The physical organization of tables and indices within the database must be disclosed.

The measured database configuration used a total of one 9Gb (common tray), 96 9Gbyte, and 48 4Gbyte Hot Swap disk drives.

2.3 Insert and Delete Operations

I*t must be ascertained that inset and delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restrictions in the SUT database 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 minimum key value for these new rows.*

All insert and delete functions were fully operational and verified during the entire benchmark.

2.4 Partitioning

While there are a 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 on any table.

2.5 Replication, Duplication or Additions

Replication of tables, if used, must be dislosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

No replications, duplications or additional attributes were used.

Section 3.0 – Clause 2 Related Items

3.1 Random Number Generation

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

The library routine SRAND48 (3C) was used to seed the library routine DRAND48 (3C) which generated pseudo-random numbers using the well-known linear congruential algorithm and 48-bit integer arithmetic. Further information on SRAND48 (3C) and DRAND48 (3C) can be found in the HP-UX Reference Manual Vol. 3.

3.2 Input/Output Screen Layout

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

The screen layouts corresponded exactly to those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C® Standard Specification.

3.3 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).

The terminal features were verified by manually exercising each specification on a NetServer system running a browser which verified the web interface.

3.4 Presentation Manager 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. Used capabilities of the terminal beyond basic ASCII entry and display were restricted to cursor positioning.

A presentation manager was not used.

3.5 Transaction Statistics

Table 2.3 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

Table 3.1: Transaction Statistics

Type	Item	Value
New Order	Home warehouse items	90.48%
	Remote warehouse items	9.52%
	Rolled back transactions	1.01%
	Average items per order	10.00
Payment	Home warehouse	85.02%
	Remote Warehouse	14.98
	Non-primary key access	60.04
Order Status	Non primary key access	59.54
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.83%
	Payment	43.06%
	Order Status	4.05%
	Delivery	4.03%
	Stock Level	4.03%

3.6 Queueing Mechanism

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

Delivery transactions were submitted to servers using the microsoft tpcc kit mechanism (delivery.exe). The difference was that the call was asynchronous, i.e., control would return to the client process immediately and the deferred delivery part would complete asychronously.

Section 4.0 – Clause 3 Related Items

4.1 Transaction System Properties (ACID Tests)

Results of the ACID test must describe how the 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). The following subsections will define each of these properties and describe the series of tests that were performed by HP to demonstrate that the properties were met.

All of the specified ACID tests were performed on the HP NetServer LXr Pro8. A fully scaled database was used except for the durability tests of durable media failure. The test was performed on a database scaled to 10 warehouses, using the standard driving mechanism. However a fully scaled database under a full load would also pass this durability test.

4.2 Atomicity Tests

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

4.2.1 COMMIT Transaction

The following steps were done to demonstrate the COMMIT property of Atomicity:

1. A row was randomly selected from the Warehouse, District and Customer tables, and the present balances noted

2. The standard payment transaction was started against the above identifiers using a known amount.

3. The transaction was committed and the rows were verified to contain the correct updated balances.

4.2.2 ROLLBACK Transaction

The following steps were done to demonstrate the ROLLBACK property of Atomicity:

1. A row was randomly selected from the Warehouse, District, Customer tables, and the present balances noted.

2. The standard payment transaction was started against the above identifiers using a known amount.

3. The transaction was rolled back and the rows were verified to contain the original balances.

4.3 Consistency Tests

Consistency is the property of the application that requires any execution of the transaction to take the database from one consistent state to another.

To prove consistency, queries were issued to the database. The results of the queries verified that the database was consistent for all conditions as specified in clause 3.3.2.1 to 3.3.2.4.

The consistency tests were run before and after the performance run.

4.4 Isolation Tests

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

This property is commonly called serializability. Sufficient conditions must be enabled at either the system or application level to ensure serializability of transactions under any mix of arbitrary transactions.

We ran a total of nine isolation tests. Seven of these tests are detailed in the TPC-C specification (clause 3.4.2.1to 3.4.2.7). The additional two are to fully comply with the isolation requirements that are not directly specified in the TPC-C specification. These two tests are known as Phantom Protection One and Two. They demonstrate that the applications are protected from phantom inserts.

4.5 Durability Tests

The tested system must guarantee the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in clause 3.5.3.1, 3.5.3.2, and 3.5.3.3.

There 3 types of failures were tested to ensure the durability of the database: Loss of Data drive, Loss of Log drive, and Loss of Memory test.

A fully scaled database was used for the Loss of Memory and the Loss of Log test while a 10 warehouse database was used for the Loss of Data test. With this exception of scaling, all other aspects of the configurations on the 10 warehouse database were identical to the fully scaled database configuration, including the use of the standard RTE drivers. Given this, the Loss of Data test would pass in a fully scaled database configuration.

TESTING PROCEDURE AND RESULTS:

The following steps detail the testing procedure and results for all the three durability tests. Each test was done separately.

Step 1: Database was backed up.

Step 2: The total number of new orders was calculated and recorded. Consistency test #3 was run to show that the database was in a consistent state prior to the durability tests.

Step 3: The standard TPC-C benchmark was launched. For the Loss of Data test, the benchmark was run with 100 users. The transaction rate was monitored until the system was in steady state. During this time, the number of users in the benchmark run was verified. After this, a checkpoint was issued. An additional 3-minute run was performed.

Step 4:The failure was initiated. For the Loss of Data drive test, one HP 4Gb Hot Swap drive holding a portion of the database data was pulled out while the benchmark was running. For the Loss of Log drive test, one HP 4Gb Hot Swap drive holding a portion of the mirrored database log was pulled out while the benchmark was running. For the Loss of Memory test, the power switch on the NetServer LXr Pro8 was depressed (turning off the system) while the benchmark was running.

Step 5: The recovery process was performed.

For the loss of Data drive test, we then backed up the transaction log, and restored the combination of the initial back up (step 1) and the just-backed-up transaction log to bring it to the most recent consistent state.

For the loss of log test, as would be expected, Mylex EDAC produced an alert message informing us that one of the members of the mirror set has failed. The SUT slowed down for a brief period as it needs to alter its log-write destination to the primary drives of the mirror. These activities were transparent to the database server as we observed that it continued to run after the aforementioned slow-down period.

For the loss of memory test, we re-powered the system, and started the server. As we would have expected, the server performed the automatic recovery.

Step 6: We computed the total number of order transactions again, and the difference between it and the one measured in step two. We verified that this difference was the same as the total number of new order transactions recorded in the “success” file. This file records committed transactions on the clients. In addition, we reran the consistency test #3 to show the database was in a consistent state after the durability tests.

We sampled the after-failure database with those recorded in the “success” file. We chose the first, last and middle two transactions from the “success” file to sample the database.

Section 5.0 – Clause 4 Related Items

5.1 Database Layout

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

The measured (tested) and priced system have identical configurations. Both configurations used two SCSI-2 Fast/Wide PCI Disk controllers that were embedded onto the motherboard and 6 Mylex DAC960-PJ 3-channel PCI Disk Array Controllers (DACs). These cards plugged into PCI slots on the motherboard.

One HP 9Gb HP Fast SCSI-2 hard disk (common tray) and one CDROM drive were attached to the first (A) of the two embedded PCI SCSI controllers. The 9Gb drive was used for the Operating System (NT v4.0).

For the both configurations a total of 10 9GB HP SCSI-2 Hot Swap hard disks are used to supply growth space for the log. A Mylex EDAC was connected to the internal SCSI controller which had two sets of five 9GB disk drives hardware mirrored 96 HP 9Gbyte Hot Swap drives are attached to 4 of the HP NetRAID PCI Disk Array controllers and 48 HP 4Gbyte Hot Swap drives are attached to the other two controllers. Eight Hot Swap disks were placed in each HP Rack Storage 8. Each channel was striped using the Daccf Utility and channel spanning was used. Controller write-back caching and read ahead were specifically disabled.

At the operating system, NT's disk administrator shows 8 logical disks - the 9Gbyte SCSI-2 boot drive (common tray), the hardware mirrored drive used for the log, two 97GB logical drive disks and four 208 GB logical drive disks. Each of these 97GB logical drives represent a hardware stripe set of twenty-four 4Gbyte Hot Swap drives, created at the DAC level spaning the three channels. The 208Gbyte drives are the same DAC configuration, except that the hard disks are 9GB each. Protection against data loss from a failed drive was achieved by normal database level recovery from the log drives, which are mirrored.

5.2 Initial Cardinality of Tables

The cardinality (e.g. 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, the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed

Table 5.1: Number of Rows

Table	Occurrences
Warehouse	1,350
District	13,500
Customer	40,500,000
History	40,500,000
Orders	40,500,000
New Orders	12,150,000
Order Line	405,000,764
Stock	135,000,000
Item	100,000

No rows were deleted for the benchmark runs.

5.3 180 Day Space

Details of the 180 day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynmic tables must be disclosed.

Transaction Log Space Requirements

To calculate the space required to sustain the database log for 8 hours of growth at steady state, the following steps were followed:

1. The free space on the logfile was queried using **dbcc checktable(syslogs)**.

2. Transactions were run against the database with a full load of users.

3. The free space was again queried using **dbcc checktable(syslogs**)

4. The space used was calculated as the difference between the first and second query.

5. The number of NEW-ORDERS was verified from an RTE report covering the entire run.

6. The space used was divided by the number of NEW-ORDERS giving a space used per NEW-ORDER transaction.

7. The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

The result of the above steps yielded a requirement of 42.9GB (including mirror) to sustain the log for 8 hours. Space in the measured and priced configurations

available on the transaction log was 43.3GB (including mirror), indicating enough storage was configured to sustain 8 hour growth.

The same methodology was used to calculate the growth requirements for the other dynamic tables Order, Order-Line and History.

The details of the 180 day growth calculation are shown in appendix D.

5.4 Type of Database Used

A statement must be provided that describes 1) the data model implemented by DBMS used and 2) the database interface and access language

Microsoft SQL Server 6.5 is a relational DBMS.

The interface was SQL Server stored procedures accessed with library calls embedded in C code.

5.5 Database Mapping

The mapping of database partitions and replications must be described.

The database was neither partitioned nor replicated.

Section 6.0 – Clause 5 Related Items

6.1 Throughput

Measured tpmC® must be reported.

Table 6.1: Throughput

tpmC®	16,257.20

6.2 ResponseTimes

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

Table 6.2: Response Times

Type	Average	Maximum	90th Percentile
New Order	0.82	4.98	1.36
Payment	0.63	4.67	1.11
Order-Status	1.74	6.49	2.79
Interactive Delivery	0.37	2.38	0.70
Deferred Delivery	0.84	3.80	1.30
Stock-Level	3.45	8.09	5.21
Menu	0.26	3.59	0.60

6.3 Keying and Think Times

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

Table 6.3: Keying Times

Type	Minimum	Average	Maximum
New-Order	18.01	18.02	18.17
Payment	3.01	3.02	3.17
Order-Status	2.01	2.02	2.16
Interactive Delivery	2.01	2.02	2.16
Stock Level	2.01	2.02	2.16

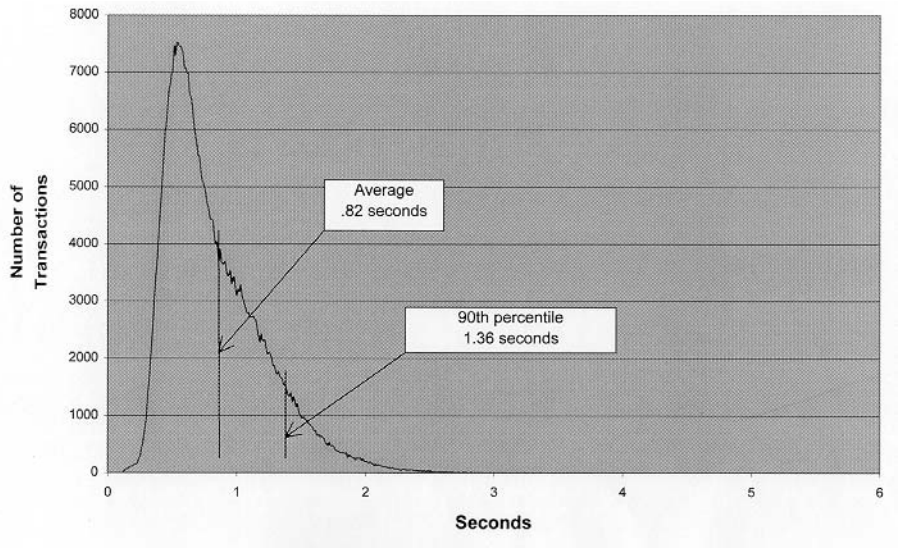
Table 6.4: Think Times

Type	Minimum	Average	Maximum
New-Order	0.01	12.13	168.66
Payment	0.01	12.17	174.52
Order-Status	0.01	10.14	107.33
Interactive Delivery	0.01	5.14	59.96
Stock-Level	0.01	5.18	65.49

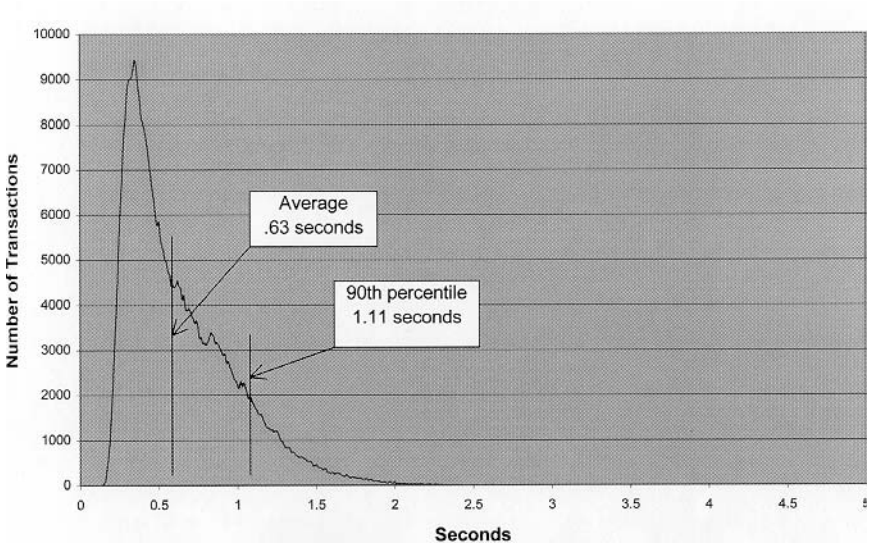
6.4 Response Time Frequency and

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type. Keying Time frequency distribution curves (see Clause 5.6.4) must be reported for each transaction type. A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction.

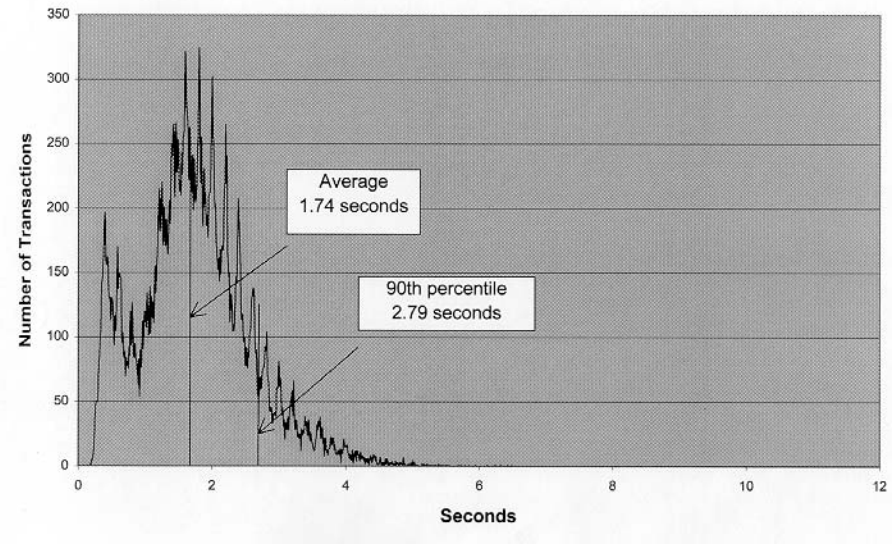
6.4.1 New Order Response Time



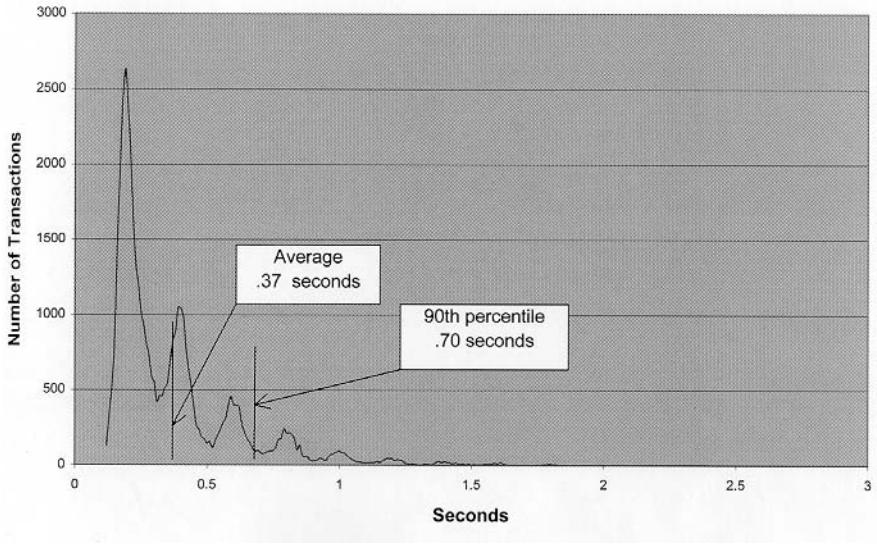
6.4.2 Payment Response Time Distribution



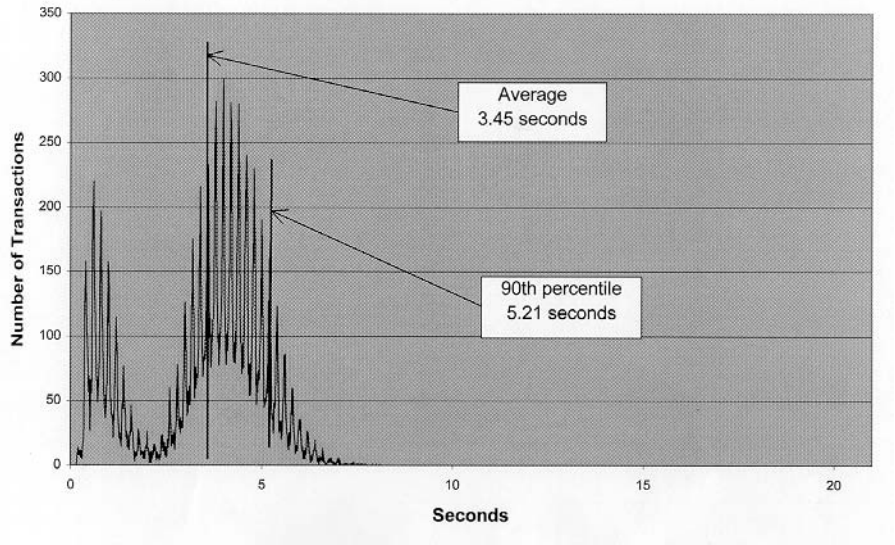
6.4.3 Order Status Response Time



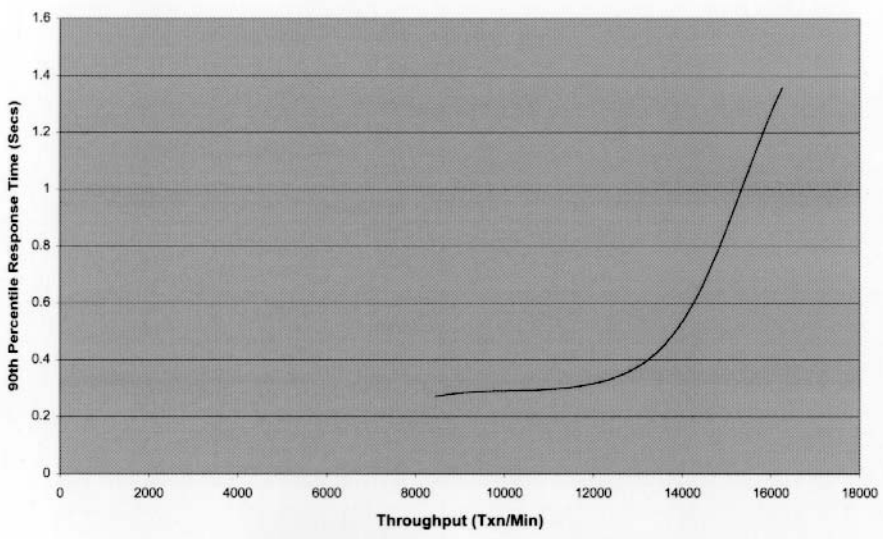
6.4.4 Delivery Response Time Distribution



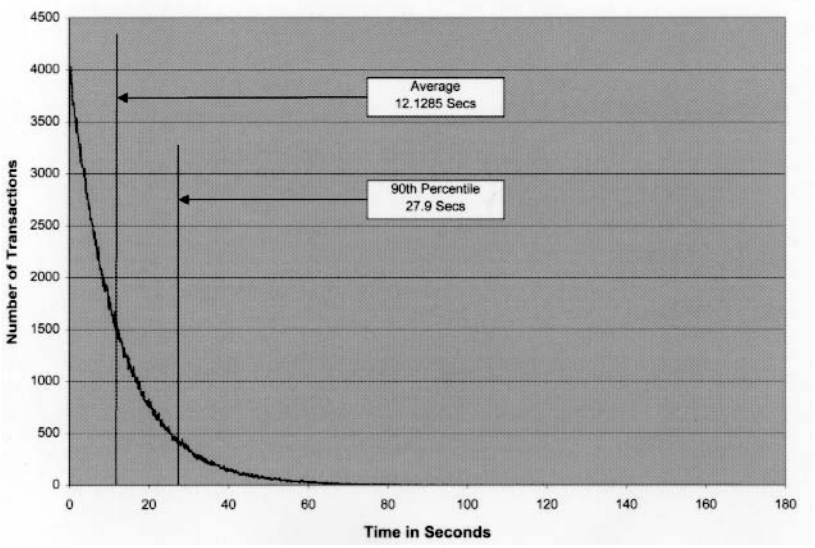
6.4.5 Stock Level Response Time



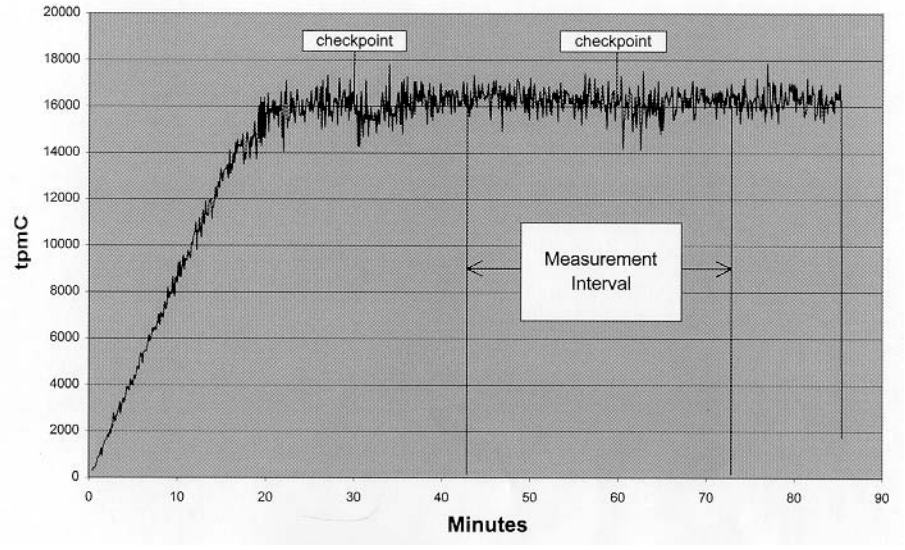
6.4.6 Response Time Versus Throughput



6.4.7 New Order Think Time Distribution



6.4.8 Throughput Versus Time Distribution



6.5 Steady State Determination

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval must be disclosed.

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 interval over the duration of the benchmark. Ramp up, steady state and ramp down regions are discernible in the graph (6.4.8).

6.6 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.

6.6.1 Checkpoint

The checkpoint mechanism is an automatic means for guaranteeing that completed transactions are regularly written from SQL Server's disk cache to the database device. A checkpoint writes all "dirty pages"-cached pages that have been modified since the last checkpoint-to the database device.

6.6.2 Checkpoint Conditions

There are two types of checkpoints:

- Checkpoints that are executed automatically by SQL Server.
- Checkpoints that are forced by database owners of the SA with the CHECKPOINT statement.

Forcing dirty pages onto the database device means that all completed transactions are written out. By calling all completed transactions to be written out, the check point shortens the time it takes to recover, since the database pages are current and there are no transactions that need to be rolled forward.

6.6.3 Checkpoint Implementation

For each benchmark measurement after all users are active, an NT command script issues a checkpoint. A background process sleeps and performs another checkpoint every 30 minutes. The recovery interval (used to control the checkpoints executed automatically by SQL Server) is configured large enough that no other checkpoints occur during the measurement.

6.7 Reproducibility

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

A second measurement achieved a throughput of 16129.77 tpmC® during a 30-minute, steady state interval.

6.8 Measurement Period 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.

6.9 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 weighted average method of *Clause 5.2.4.1* was used. The weights were not adjusted during the run.

6.10 Transaction Mix

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

Table 6.5: Transaction Mix

Type	Percentage
New-Order	44.83%
Payment	43.06%
Order-Status	4.05%
Delivery	4.03%
Stock-Level	4.03%

6.11 Transaction Statistics

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 selections made 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 2.1 contains the required items.

6.12 Checkpoint Count and Location

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.

Times in the following table are relative to the beginning of the driver-times phase of the test. The checkpoint interval is 30 minutes. The first checkpoint within the 30 minute measure interval was 1804 seconds from its start. In accord with *5.5.2.2*, there is no checkpoint within the "guard zones" 1800/4=450 seconds from the beginning and end of the measurement interval.

Table 6.6: Checkpoints

Event	From (sec)	To (sec)	Duration (sec)
checkpoint	1804	2166	362
measured interval	2616	4416	1800
checkpoint	3608	3975	367

Section 7.0 – Clause 6 Related Items

7.1 RTE description

If the RTE is commercilally available, then its inputs must be specified. Otherwise, a description must be supplied of that inpute (e.g., scripts) to the RTE had been used. The RTE input parameters, code fragments, functions, et cetera used to generate each transaction input filed must be disclosed. Comment: the t is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2. Appendix A.3 lists RTE input parameters and code fragments used to generate each transaction input field.

The RTE (remote Terminal Emulator) on the driver system was developed at Hewlett Packard and is not commericallly available.

For this instance of the TPC-C benchmark, six driver and three client systems were used. The drivers emulated 13500 users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in figure 7.1

The benchmark is started with the do_runs command on the driver system. do_runs controls the overall execution of the benchmark. After reading a configuration file, do_runs starts the TUXEDO servers on the clients, collects pre-benchmark audit information and inserts a timestamp into a database audit table. When all the initial steps are completed, do_runs invokes another program, DRIVER, to start the benchmark. Results are collected into a single location at the completion of the run.

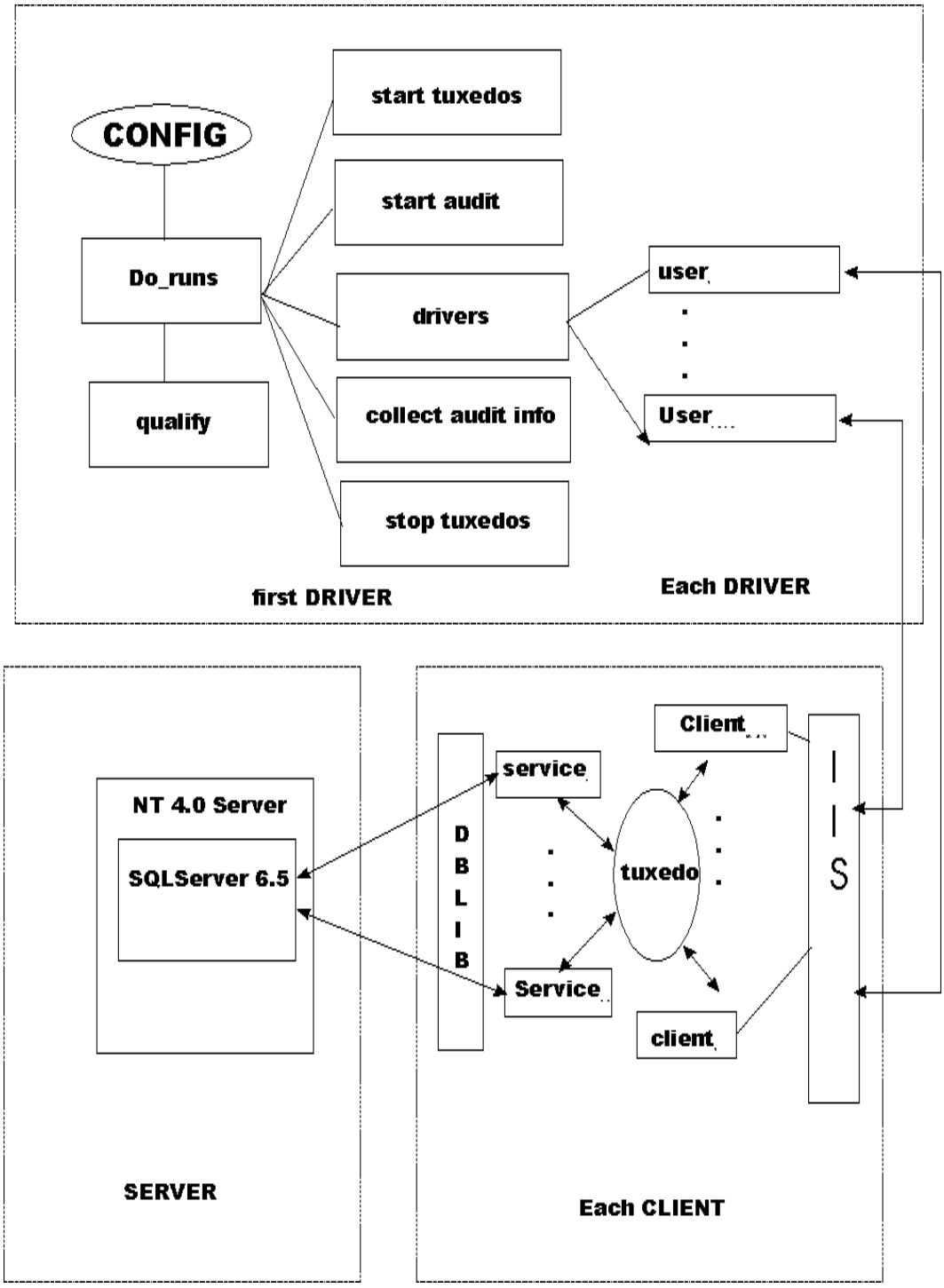
DRIVER is the heart of the benchmark software. It simulates users as they log in. execute transactions and view results. DRIVER collects response times for each transaction and saves them in a file for future analysis.

QUALIFY is the post-processing analysis program. This is executed on the master RTE machine, the contolling RTE. It produces the numerical summaries and histograms needed for the disclosure report.

Appendix A contains listings of the code used to generate the transaction input.

7.2 Emulated Components

FIGURE 7-1: Benchmark Software



7.3 Functional Diagram

A complete functional diagram of the hardware and software of the benchmark configuration including the driver must be provided. the sponsor must list all hardware and software functionality of the driver and its interface to the SUT.

Figures 1.1 and 1.2 in chapter 1 show functional diagrams of the benchmark and configured systems. A description of the RTE and benchmark software is provided above.

7.4 Networks

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

Figures 1.1 and 1.2 in chapter 1 diagram the network configurations of the benchmark and configured systems, and represent the Driver connected via LAN replacing the workstations and HUBS connected via LANs.

The bandwidth of the networks used in the tested/priced configurations must be disclosed.

Ethernet and 10 Base-T local area networks (LAN) with a bandwidth of 10 megabits per second are used in the tested/priced configurations.

Section 8.0 – Clause 7 Related Items

8.1 System 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 data. 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 and effecive date(s) of price(s) must also be reported.

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

The details of the hardware, software and maintenance components of this system are reported in the front of this report as part of the executive summary. All 3rd party quotations are included at the end of this report in Appendix E.

8.2 General Availability, Throughput, and Price Performance

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 availabilty dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

All hardware and software components of this system are currently available.

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

MAXIMUM QUALIFIED THROUGHPUT:	**16257.20 tpmC**
PRICE per tpmC:	**$33.67 per tpmC**
HARDWARE AVAILABILITY:	**January 1998**
SOFTWARE AVAILABILITY:	**January 1998**

8.3 Country Specific Pricing

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

The system is being priced for the United States of America.

8.4 Usage Pricing

For any usage pricing, the sponser must disclose 1) Usage level at which the component was priced, 2) a statement of the company policy allowing such pricing.

The component pricing based on usage is shown below:

Microsoft SQL Server v6.5 User License was priced for unlimited number of users.

9.0 Clause 9 Related Items

9.1 Auditor's Information

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.

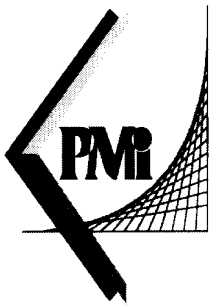
The test methodology and results of this TPC Benchmark C were audited by:

Performance Metrics, Inc
TPC Certified Auditors
2229 Benita Drive, Suite 101
Rancho Cordova, CA 95670
phone: 916 635-2822
fax: 916 858-0109

The auditor was Richard Gimarc. A copy of the Attestation Letter received from the auditor is attached on the following pages.

Requests for this Full Disclosure Report (FDR) should sent to:

Hewlett-Packard Company
Network Server Division
Attention: Kuppuswamy Sivakumar
10450 Ridgeview Court, Bldg. 49E
MS 49EL-FS
Cupertino, CA 95015-4050



PERFORMANCE METRICS INC.
TPC Certified Auditors

March 25, 1998

Misters Anh Nguyen and Kuppuswamy Sivakumar
R & D Performance Engineers
Hewlett Packard Corporation
5301 Stevens Creek Boulevard
Santa Clara, CA 95052-8059

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

Platform:	Hewlett-Packard NetServer LXr Pro 8
Database Manager:	Microsoft SQLServer 6.5 Enterprise Edition
Operating System:	Microsoft NT 4.0 SP3 Enterprise Edition
Transaction Manager:	Tuxedo 6.4 for Windows NT

Server: NetServer LXr Pro 8				
CPU's	Memory	Disks	90% Response	tpmC
8 Pentium Pro @ 200 Mhz	Main: 4 GB Cache: 1MB each	107 @ 9.1GB 48 @ 4.3 GB	1.36 sec	16,257.20
3 Clients: Hewlett-Packard LC II				
2 Pentium II @ 266 Mhz	Main: 512 MB Cache: 512K	1 @ 4.3 GB	na	na

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 1,350 warehouses.
- The ACID properties were met.
- The Durability data-loss test was performed on a database scaled to 10 warehouses.

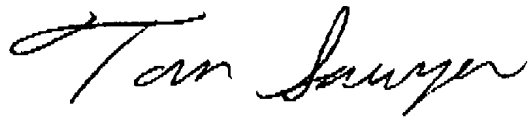
2229 Benita Dr. Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 fax: (916) 858-0109 email: Lorna@PerfMetrics.com

- Eight hours of mirrored log space was present on the tested system.
- Eight hours of growth space for the dynamic tables was present on the tested system.
- The data for the 180-day space calculation was verified. No disks were added to the priced configuration
- The steady state portion of the test was 30 minutes.
- One checkpoint was taken before the measured interval.
- One checkpoint was taken during the measured interval.
- The checkpoints were verified to be clear of the guard zone.
- The system pricing was checked for major components and maintenance.

Auditor Notes:

The majority of the audit was performed by Mr. Richard Gimarc and Ms Lorna Livingtree.

Sincerely,



Tom Sawyer
Auditor

2229 Benita Dr. Suite 101, Rancho Cordova, CA 95670
(916) 635-2822 fax: (916) 858-0109 email: Lorna@PerfMetrics.com

Appendix A – Application Source

A.1 Client Front-End

This appendix contains the source and makefiles for the Tuxedo client and server programs. All of the programs ran on the client machine. The Tuxedo version used was built using the makefile in tuxedo_threads.mak, in Appendix A.

db.h

```
1  #ifdef USE_ODBC
2  void dbsetuserdata(PDBPROCESS dbproc, void *uPtr);
3  void *dbgetuserdata(PDBPROCESS dbproc);
4  void BindParameter(PDBPROCESS dbproc, UWORD ipar,
SWORD fCType, SWORD fSqlType, UDWORD cbColDef, SWORD
ibScale, PTR rgbValue, SDWORD cbValueMax);
5  void ODBCError(PDBPROCESS dbproc);
6  BOOL ExecuteStatement(PDBPROCESS dbproc, char
*szStatement);
7  BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol,
SQLSMALLINT fCType, SQLPOINTER rgbValue, SQLINTEGER
cbValue ax);
8  BOOL GetResults(PDBPROCESS dbproc);
9  BOOL MoreResults(PDBPROCESS  dbproc);
10 BOOL ReopenConnection(PDBPROCESS dbproc);
11 #endif
```

delirpt.c

```
1  /*  FILE:       DELIRPT.C
2   *              Microsoft TPC-C Kit Ver. 3.00.000
3   *
4   *              Copyright Microsoft, 1996
5   *
6   *  PURPOSE:    Delivery report processing applica-
tion
7   *  Author:     Philip Durr
8   *              philipdu@Microsoft.com
9   */
10
11 #include <windows.h>
12 #include <stdio.h>
13 #include <stdlib.h>
14
15 #define LOGFILE_READ_EOF 0 //check log file flag
return current state
16 #define LOGFILE_CLEAR_EOF   1 //clear end of log file
flag
17 #define LOGFILE_SET_EOF     2 //set flag end of log
file reached
18
19 #define INTERVAL .01 //90th percentile calculation
bucket interval
20
21 #define ERR_SUCCESS 1000 //success no error
22 #define ERR_READING_LOGFILE 1001 //io errors occured
reading delivery log file
23 #define ERR_INSUFFICIENT_MEMORY 1002 //insuficient
memory to process 90th percentile report
24 #define ERR_CANNOT_OPEN_RESULTS_FILE 1005 //Cannot
open delivery results file delilog.
25
26 typedef struct _RPTLINE
27 {
28 SYSTEMTIME start; //delilog report line start time
29 SYSTEMTIME  end; //delilog report line end time
30 int response; //delilog report line time delivery
took in milliseconds
31 int w_id; //delilog report line warehouse id for
delivery
32 int o_carrier_id; //delilog report line carier id for
delivery
33 int items[10];//delilog report line delivery line
items
34 } RPTLINE, *PRPTLINE;
35
36 //error message structure used in ErrorMessage API
37 typedef struct _SERRORMSG
38 {
39 int iError; //error id of message
40 char szMsg[80]; //message to sent to browser
41 } SERRORMSG;
42
43 int versionMS = 4; //delirpt version
44 int             versionMM = 0;
45 int             versionLS = 0;
46 int             iReport;
//delirpt report to process
47 int             iStartTime;
//begin times to accept for report
48 int             iEndTime;
//end times to accept for report
49 FILE            *fpLog;
//log file stream
50
51 //Local function prototypes
52 void        main(int argc, char *argv[]);
53 static int  Init(void);
54 static void Restore(void);
55 static int  DoReport(void);
56 int         AverageResponse(void);
57 int         SkippedDelivery(void);
58 int         Percentile90th(void);
59 BOOL        CheckTimes(PRPTLINE pRptLine);
60 static int  OpenLogFile(void);
61 static void CloseLogFile(void);
62 static void ResetLogFile(void);
63 static BOOL LogEOF(int iOperation);
64 static BOOL ReadReportLine(char *szBuffer, PRPTLINE
pRptLine);
65 static BOOL ParseReportLine(char *szLine, PRPTLINE
pRptLine);
66 static BOOL ParseDate(char *szDate, LPSYSTEMTIME
pTime);
67 static BOOL ParseTime(char *szTime, LPSYSTEMTIME
pTime);
68 static void ErrorMessage(int iError);
69 static BOOL GetParameters(int argc, char *argv[]);
70 static void PrintParameters(void);
71 static void PrintHeader(void);
72 static void cls(void);
73 static BOOL IsNumeric(char *ptr);
74
75
76 /* FUNCTION: int main(int argc, char *argv[])
77  *
78  * PURPOSE: This function is the beginning execution
point for the delivery executable.
79  *
80  * ARGUMENTS:   int    argc    number of command line
arguments passed to delivery
81  *              char    *argv[] array of command line
argument pointers
82  *
83  * RETURNS:     None
84  *
85  * COMMENTS:    None
86  *
87  */
88
89 void main(int argc, char *argv[])
90 {
91     int iError;
```

```
92
93      PrintHeader();
94
95      if ( GetParameters(argc, argv) )
96      {
97          PrintParameters();
98          return;
99      }
100
101     if ( (iError=Init()) != ERR_SUCCESS )
102     {
103         ErrorMessage(iError);
104         Restore();
105         return;
106     }
107
108     if ( (iError = DoReport()) != ERR_SUCCESS )
109         ErrorMessage(iError);
110
111     Restore();
112
113     return;
114 }
115
116 /* FUNCTION: static int Init(void)
117  *
118  * PURPOSE: This function initializes the delirtp
application.
119  *
120  * ARGUMENTS:   None
121  *
122  * RETURNS:     None
123  *
124  * COMMENTS:    None
125  *
126  */
127
128 static int Init(void)
129 {
130     int iError;
131
132     if ( (iError = OpenLogFile()) )
133         return iError;
134     return TRUE;
135 }
136
137 /* FUNCTION: static void Restore(void)
138  *
139  * PURPOSE: This function cleans up the delirpt
application before termination.
140  *
141  * ARGUMENTS:   None
142  *
143  * RETURNS:     None
144  *
145  * COMMENTS:    None
146  *
147  */
148
149 static void Restore(void)
150 {
151     CloseLogFile();
152     return;
153 }
154
155 /* FUNCTION: static int DoReport(void)
156  *
157  * PURPOSE:     This function dispatches the
requested report.
158  *
159  * ARGUMENTS:   None
160  *
161  * RETURNS:     ERR_SUCCESS if successfull or error
code if an error occurs.
162  *
163  * COMMENTS:    None
164  *
165  */
166
167 static int DoReport(void)
168 {
169     int iRc;
170
171     switch(iReport)
172     {
173         case 1:
174             iRc = AverageResponse();
175             break;
176         case 2:
177             iRc = Percentile90th();
178             break;
179         case 3:
180             iRc = SkippedDelivery();
181             break;
182         case 4:
183             if ( (iRc = AverageResponse()) !=
ERR_SUCCESS )
184                 break;
185             if ( (iRc = Percentile90th()) !=
ERR_SUCCESS )
186                 break;
187             if ( (iRc = SkippedDelivery()) !=
ERR_SUCCESS )
188                 break;
189             break;
190     }
191     return iRc;
192 }
193
194 /* FUNCTION: int AverageResponse(void)
195  *
196  * PURPOSE:     This function processes the Average-
Response report.
197  *
198  * ARGUMENTS:   None
199  *
200  * RETURNS:     ERR_SUCCESS if successfull or error
code if an error occurs.
201  *
202  * COMMENTS:    None
203  *
204  */
205
206 int AverageResponse(void)
207 {
208     RPTLINE reportLine;
209     int     iTotalResponse;
210     int     iLines;
211     double  fAverage;
212     char    szDelivery[128];
213
214     ResetLogFile();
215
216     iTotalResponse = 0;
217     iLines = 0;
218    printf("\n\n******** Average Response Time Report
*******\n");
219     while ( !LogEOF(LOGFILE_READ_EOF) )
220     {
221         if ( ReadReportLine(szDelivery, &reportLine)
)
222             return ERR_READING_LOGFILE;
223         if ( !LogEOF(LOGFILE_READ_EOF) )
224         {
225             if ( CheckTimes(&reportLine) )
226                 continue;
227             iLines++;
228             iTotalResponse += reportLine.response;
229
230             if ( iLines % 10 == 0 )
231                 printf("Reading Report Line:\t%d\r",
iLines);
232         }
233     }
```

```
234     printf("
\r");
235     if ( iLines == 0 )
236     {
237         printf("No deliveries found.\n");
238     }
239     else
240     {
241         fAverage = ((double)iTotalResponse / (dou-
ble)iLines)/(double)1000;
242         printf("Total Deliveries:      %10.0f\n",
(float)iLines);
243         printf("Total Response Times:  %10.3f\n",
((float)iTotalResponse/(float)1000));
244         printf("Average Response Time: %10.3f\n",
fAverage);
245     }
246
247     return ERR_SUCCESS;
248 }
249
250 /* FUNCTION: int Percentile90th(void)
251  *
252  * PURPOSE:     This function processes the 90th per-
centile report.
253  *
254  * ARGUMENTS:   None
255  *
256  * RETURNS:     ERR_SUCCESS if successfull or error
code if an error occurs.
257  *
258  * COMMENTS:    This function requires enough space
to allocate needed
259  *              buckets which will be 2 * max response
time in
260  *              deci-seconds.
261  *
262  */
263
264 int Percentile90th(void)
265 {
266     RPTLINE reportLine;
267     int     iBucketSize;
268     int     i;
269     int     iResponseSeconds;
270     int     iMaxSeconds;
271     int     iTotalBuckets;
272     double  iTotal;
273     double  i90thPercent;
274     short   *psBuckets;
275     char    szDelivery[128];
276
277     printf("\n\n******** 90th Percentile *******\n");
278     printf("Calculating Max Response Seconds...\n");
279
280     ResetLogFile();
281
282     iMaxSeconds = -1;
283     while ( !LogEOF(LOGFILE_READ_EOF) )
284     {
285         if ( ReadReportLine(szDelivery, &reportLine)
)
286             return ERR_READING_LOGFILE;
287         if ( szDelivery[0] == '*' )
288             continue;
289         if ( !LogEOF(LOGFILE_READ_EOF) )
290         {
291             if ( iMaxSeconds < reportLine.response )
292                 iMaxSeconds = reportLine.response;
293         }
294     }
295
296     iTotalBuckets = iMaxSeconds + 1;
297
298     printf("Allocating Buckets...\n");
299
300     iBucketSize = iTotalBuckets * sizeof(short);
301
302     if ( !(psBuckets = (short *)malloc(iBucketSize))
)
303         return ERR_INSUFFICIENT_MEMORY;
304
305     ZeroMemory(psBuckets, iBucketSize);
306
307     iTotal = 0;
308
309     ResetLogFile();
310     printf("Calculating Distribution...\n");
311
312     while ( !LogEOF(LOGFILE_READ_EOF) )
313     {
314         if ( ReadReportLine(szDelivery, &reportLine)
)
315             return ERR_READING_LOGFILE;
316         if ( szDelivery[0] == '*' )
317             continue;
318         if ( !LogEOF(LOGFILE_READ_EOF) )
319         {
320             if ( CheckTimes(&reportLine) )
321                 continue;
322             psBuckets[reportLine.response]++;
323             iTotal++;
324         }
325     }
326
327     i90thPercent = iTotal * .9;
328
329     for(i=0, iTotal = 0.0; iTotal < i90thPercent;
iTotal += (double)psBuckets[i] )
330         i++;
331
332     printf("90th Percentile = %d.%d\n", i/1000, (i %
1000));
333
334     free(psBuckets);
335
336     return ERR_SUCCESS;
337 }
338
339 /* FUNCTION: int SkippedDelivery(void)
340  *
341  * PURPOSE:     This function processes the Skipped
Deliveries report.
342  *
343  * ARGUMENTS:   None
344  *
345  * RETURNS:     ERR_SUCCESS if successfull or error
code if an error occurs.
346  *
347  * COMMENTS:    None
348  *
349  */
350
351 int SkippedDelivery(void)
352 {
353     RPTLINE reportLine;
354     char    szDelivery[128];
355     int     i;
356     int     items[10];
357
358     ResetLogFile();
359
360     printf("\n\n******** Skipped Delivery Report
*******\n");
361     memset(items, 0, sizeof(items));
362     printf("Reading Delivery Log File...");
363
364     while ( !LogEOF(LOGFILE_READ_EOF) )
365     {
366         if ( ReadReportLine(szDelivery, &reportLine)
)
367             return ERR_READING_LOGFILE;
368         if ( !LogEOF(LOGFILE_READ_EOF) )
369         {
```

```
370             if ( CheckTimes(&reportLine) )
371                 continue;
372             for(i=0; i<10; i++)
373             {
374                 if ( !reportLine.items[i] )
375                     items[i]++;
376             }
377         }
378     }
379     printf("\n");
380     printf("Skipped delivery table.\n");
381    printf("  1    2    3    4    5    6    7    8    9
10 \n");
382     printf("---- ---- ---- ---- ---- ---- ---- ---- -
--- ----\n");
383     for(i=0; i<10; i++)
384         printf("%4.4d ", items[i]);
385     printf("\n");
386
387     return ERR_SUCCESS;
388 }
389
390 /* FUNCTION: BOOL CheckTimes(PRPTLINE pRptLine)
391  *
392  * PURPOSE: This function checks to see of the
delilog record falls withing the
393  *          begin and end time from the command line.
394  *
395  * ARGUMENTS:   PRPTLINE    pRptLine    delilog pro-
cessed report line.
396  *
397  * RETURNS:     BOOL    FALSE   if report line is not
within the
398  *                              requested start and
end times.
399  *                      TRUE    if the report line is
within the
400  *                              requested start and
end times.
401  *
402  * COMMENTS:    If startTime and endTime are both 0
then the user requested
403  *              the default behavior which is all
records in delilog are
404  *              valid.
405  */
406
407 BOOL CheckTimes(PRPTLINE pRptLine)
408 {
409     int iRptEndTime;
410     int iRptStartTime;
411
412    iRptStartTime = (pRptLine->start.wHour * 3600000)
+ (pRptLine->start.wMinute * 60000) + (pRptLine-
>start.wSecond * 1000) + pRptLine->start.wMilliseconds;
413     iRptEndTime = (pRptLine->end.wHour * 3600000) +
(pRptLine->end.wMinute * 60000) + (pRptLine->end.wSecond
* 1000) + pRptLine->end.wMilliseconds;
414
415     if ( iStartTime == 0 && iEndTime == 0 )
416         return FALSE;
417
418     if ( iStartTime <= iRptStartTime && iEndTime >=
iRptEndTime )
419         return FALSE;
420
421     return TRUE;
422 }
423
424 /* FUNCTION: int OpenLogFile(void)
425  *
426  * PURPOSE: This function opens the delivery log file
for use.
427  *
428  * ARGUMENTS:   None
429  *
430  * RETURNS:     int ERR_CANNOT_OPEN_RESULTS_FILE
Cannot create results log file.
431  *                 ERR_SUCCESS                     Log
file successfully opened
432  *
433  *
434  * COMMENTS:    None
435  *
436  */
437
438 static int OpenLogFile(void)
439 {
440     fpLog = fopen("delilog.", "rb");
441
442     if ( !fpLog )
443         return ERR_CANNOT_OPEN_RESULTS_FILE;
444
445     return ERR_SUCCESS;
446 }
447
448 /* FUNCTION: int CloseLogFile(void)
449  *
450  * PURPOSE: This function closes the delivery log
file.
451  *
452  * ARGUMENTS:   None
453  *
454  * RETURNS:     None
455  *
456  * COMMENTS:    None
457  *
458  */
459
460 static void CloseLogFile(void)
461 {
462     if ( fpLog )
463         fclose(fpLog);
464
465     return;
466 }
467
468
469 /* FUNCTION: static void ResetLogFile(void)
470  *
471  * PURPOSE: This function prepares the delilog. file
for reading
472  *
473  * ARGUMENTS:   None
474  *
475  * RETURNS:     None
476  *
477  * COMMENTS:    None
478  *
479  */
480
481 static void ResetLogFile(void)
482 {
483     fseek(fpLog, 0L, SEEK_SET);
484     LogEOF(LOGFILE_CLEAR_EOF);
485
486     return;
487 }
488
489 /* FUNCTION: static BOOL LogEOF(int iOperation)
490  *
491  * PURPOSE: This function tracks and reports the end
of file condition
492  *          on the delilog file.
493  *
494  * ARGUMENTS:   int iOperation  requested operation
this can be:
495  *                                 LOGFILE_READ_EOF
check log file flag return current state
496  *                                 LOGFILE_CLEAR_EOF
clear end of log file flag
497  *                                 LOGFILE_SET_EOF
set flag end of log file reached
498  *
```

```
 *
 * 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 struc-
ture 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 con-
taining the delilog file line to be parsed.
 *             PRPTLINE    pRptLine    returned struc-
ture 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 con-
taining 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 con-
taining 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->wMillisec-
onds/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 dis-
played
 *
 * 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 file
delilog."                    },
         {   ERR_READING_LOGFILE,
"Reading delivery log file, Delivery item format incor-
rect."    },
         {   ERR_INSUFFICIENT_MEMORY,
"insufficient memory to process 90th percentile report."
},
    {  0,                "\"\"
}
    };

    for(i=0; errorMsgs[i].szMsg[0]; i++)
```

```
760    {
761        if ( iError == errorMsgs[i].iError )
762        {
763            printf("\nError(%d): %s", iError,
errorMsgs[i].szMsg);
764            return;
765        }
766    }
767    printf("Error(%d): %s", errorMsgs[0].szMsg);
768    return;
769}
770
771/* FUNCTION: BOOL GetParameters(int argc, char
*argv[])
772 *
773 * PURPOSE: This function parses the command line
passed in to the delivery executable, initializing
774 *          and filling in global variable parameters.
775 *
776 * ARGUMENTS:   int     argc    number of command line
arguments passed to delivery
777 *              char    *argv[] array of command line
argument pointers
778 *
779 * RETURNS:     BOOL    FALSE   parameter read suc-
cessfull
780 *                      TRUE    user has requested
parameter information screen be displayed.
781 *
782 * COMMENTS:    None
783 *
784 */
785
786static BOOL GetParameters(int argc, char *argv[])
787{
788    int         i;
789    SYSTEMTIME  startTime;
790    SYSTEMTIME  endTime;
791
792    iStartTime = 0;
793    iEndTime = 0;
794    iReport = 4;
795
796    for(i=0; i<argc; i++)
797    {
798        if ( argv[i][0] == '-' || argv[i][0] == '/' )
799        {
800            switch(argv[i][1])
801            {
802                case 'S':
803                case 's':
804                    if ( ParseTime(argv[i]+2, &start-
Time) )
805                        return TRUE;
806                    iStartTime = (startTime.wHour *
3600000) + (startTime.wMinute * 60000) + (start-
Time.wSecond * 1000) + startTime.wMilliseconds;
807                    break;
808                case 'E':
809                case 'e':
810                  if ( ParseTime(argv[i]+2, &endTime)
)
811                        return TRUE;
812                    iEndTime = (endTime.wHour *
3600000) + (endTime.wMinute * 60000) + (endTime.wSecond
* 1000) + endTime.wMilliseconds;
813                    break;
814                case 'R':
815                case 'r':
816                    iReport = atoi(argv[i]+2);
817                    if ( iReport > 4 || iReport < 1 )
818                        iReport = 4;
819                    break;
820                case '?':
821                    return TRUE;
822            }
823        }
824    }
825    return FALSE;
826}
827
828/* FUNCTION: void PrintParameters(void)
829 *
830 * PURPOSE: This function displays the supported com-
mand line flags.
831 *
832 * ARGUMENTS:   None
833 *
834 * RETURNS:     None
835 *
836 * COMMENTS:    None
837 *
838 */
839
840static void PrintParameters(void)
841{
842    PrintHeader();
843    printf("DELIRPT:\n\n");
844    printf("Parame-
ter
Default\n");
845    printf("----------------------------------------
------------------------------\n");
846    printf("-S Start Time
HH:MM:SS:MMM                                  All
\n");
847    printf("-E End Time
HH:MM:SS:MMM                                   All
\n");
848    printf("-R 1)Average Response, 2)90th 3) Skipped
4) All                 All    \n");
849    printf("-? This help screen\n\n");
850    printf("Note:  Command line switches are NOT case
sensitive.\n");
851
852    return;
853}
854
855/* FUNCTION: void PrintHeader(void)
856 *
857 * PURPOSE: This function displays the delivery
report applications banner information.
858 *
859 * ARGUMENTS:   None
860 *
861 * RETURNS:     None
862 *
863 * COMMENTS:    None
864 *
865 */
866
867static void PrintHeader(void)
868{
869    cls();
870
871
printf("*********************************************
*\n");
872
printf("*
*\n");
873    printf("*  Microsoft SQL Server
6.5                     *\n");
874
printf("*
*\n");
875    printf("*  HTML TPC-C BENCHMARK KIT: Delivery
Report    *\n");
876    printf("*  Version
%d.%2.2d.%3.3d                          *\n", versionMS,
versionMM, versionLS);
877
printf("*
*\n");
```

```
878
printf("************************************************
*\n\n");
879
880     return;
881}
882
883/* FUNCTION: void cls(void)
884 *
885 * PURPOSE: This function clears the console window
886 *
887 * ARGUMENTS:   None
888 *
889 * RETURNS:     None
890 *
891 * COMMENTS:    None
892 *
893 */
894
895static void cls(void)
896{
897     HANDLE  hConsole;
898     COORD   coordScreen = { 0, 0 };         //here's
where we'll home the cursor
899     DWORD   cCharsWritten;
900     CONSOLE_SCREEN_BUFFER_INFO  csbi;       //to get
buffer info
901     DWORD                       dwConSize;  //number
of character cells in the current buffer
902
903     hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
904
905     //get the number of character cells in the cur-
rent buffer
906
907     GetConsoleScreenBufferInfo( hConsole, &csbi );
908     dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
909
910     //fill the entire screen with blanks
911     FillConsoleOutputCharacter( hConsole, (TCHAR) '
', dwConSize, coordScreen, &cCharsWritten );
912     GetConsoleScreenBufferInfo( hConsole, &csbi );
913
914     //now set the buffer's attributes accordingly
915     FillConsoleOutputAttribute( hConsole, csbi.wAt-
tributes,dwConSize, coordScreen, &cCharsWritten );
916
917     //put the cursor at (0, 0)
918     SetConsoleCursorPosition( hConsole, coordScreen
);
919
920     return;
921}
922
923/* FUNCTION: BOOL IsNumeric(char *ptr)
924 *
925 * PURPOSE: This function determines if a string is
numeric. It fails if any characters other
926 *          than numeric and null terminator are
present.
927 *
928 * ARGUMENTS:   char            *ptr    pointer to
string to check.
929 *
930 * RETURNS:     BOOL    FALSE   if string is not all
numeric
931 *                      TRUE    if string contains only
numeric characters i.e. '0' - '9'
932 *
933 * COMMENTS:    A comma is counted as a valid delim-
iter.
934 *
935 */
936
937static BOOL IsNumeric(char *ptr)
938{
939     if ( *ptr == 0 )
940         return FALSE;
941
942     while( *ptr && isdigit(*ptr) )
943         ptr++;
944     if ( !*ptr || *ptr == ',' )
945         return TRUE;
946     else
947         return FALSE;
948}
```

delisrv.c

```
 1  /*  FILE:       DELISRV.C
 2     *              Microsoft TPC-C Kit Ver. 3.00.000
 3     *              Audited 08/23/96, By Francois Raab
 4     *
 5     *              Copyright Microsoft, 1996
 6     *
 7     *  PURPOSE:    Delivery TPC-C transaction execut-
able
 8     *  Author:     Philip Durr
 9     *              philipdu@Microsoft.com
10     */
11
12    #include <windows.h>
13    #include <process.h>
14    #include <stdio.h>
15    #include <stdarg.h>
16    #include <malloc.h>
17    #include <stdlib.h>
18    #include <string.h>
19    #include <time.h>
20    #include <sys\timeb.h>
21    #include  <io.h>
22    #include <conio.h>
23    #include <ctype.h>
24
25    #ifdef USE_ODBC
26        #include <sql.h>
27        #include <sqlext.h>
28        HENV    henv;
29    #else
30        #define DBNTWIN32
31        #include <sqlfront.h>
32        #include <sqldb.h>
33    #endif
34
35    #include "delisrv.h"
36
37    char                    szServer[32];
      //SQL server name
38    char                    szDatabase[32];
      //tpcc database name
39    char                    szUser[32];
      //user name
40    char                    szPassword[32];
      //user password
41    int                     iNumThreads     = 4;
      //number of threads to create
42    int                     iDelayMs        = 1000;
      //delay between delivery queue checks
43    int                     iDeadlockRetry  = 3;
      //number of read check retries.
44    int                     iQSlotts        = 3000;
      //delivery transaction queues
45    int                     iConnectDelay   = 500;
      //delay between re-connect attempts if sql server
      refuses connection.
46
47    FILE                    *fpLog;
      //pointer to log file
48    CRITICAL_SECTION        WriteLogCriticalSec-
tion;    //critical section for delivery write log
49    CRITICAL_SECTION        DeliveryCriticalSec-
tion;    //critical section for delivery transactions
      cache
50    static  LPTSTR          lpszPipeName =
```

```
TEXT("\\\\.\\pipe\\DELISRV");    //delivery pipe name
51
52    HANDLE                  hPipe       =
INVALID_HANDLE_VALUE; //delivery pipe handle
53    HANDLE                  hComPort    =
INVALID_HANDLE_VALUE; //delivery pipe completion port
handle.
54
55    BOOL
bDone;                            //delivery executable
termination request flag
56    BOOL
bFlush;                     //Flush delivery log info
when written.
57
58    LPDELIVERY_PACKET       pDeliveryCache;
59
60    int                     versionMS =
4;                  //delivery executable version num-
ber.
61    int                     versionMM =
0;                  //formatted as MS.MM.LS, 1.00.005
62    int                     versionLS = 0;
63
64    /* FUNCTION: int main(int argc, char *argv[])
65     *
66     * PURPOSE: This function is the beginning execu-
tion point for the delivery executable.
67     *
68     * ARGUMENTS:   int     argc    number of command
line arguments passed to delivery
69     *              char    *argv[] array of command
line argument pointers
70     *
71     * RETURNS:     None
72     *
73     * COMMENTS:    None
74     *
75     */
76
77    void main(int argc, char *argv[])
78    {
79        int iError;
80
81        if ( GetParameters(argc, argv) )
82        {
83            PrintParameters();
84            return;
85        }
86
87        if ( (iError=Init()) )
88        {
89            ErrorMessage(iError);
90            Restore();
91            return;
92        }
93
94        if ( (iError = RunDelivery()) != ERR_SUCCESS )
95            ErrorMessage(iError);
96
97        Restore();
98
99        return;
100   }
101
102   /* FUNCTION: void cls(void)
103    *
104    * PURPOSE: This function clears the console win-
 dow
105    *
106    * ARGUMENTS:   None
107    *
108    * RETURNS:     None
109    *
110    * COMMENTS:    None
111    *
112    */
113
114   static void cls(void)
115   {
116       HANDLE  hConsole;
117       COORD   coordScreen = { 0, 0 };
 //here's where we'll home the cursor
118       DWORD   cCharsWritten;
119       CONSOLE_SCREEN_BUFFER_INFO  csbi;       //to
 get buffer info
120       DWORD                     dwConSize;  //number
 of character cells in the current buffer
121
122       hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
123
124       //get the number of character cells in the
 current buffer
125
126       GetConsoleScreenBufferInfo( hConsole, &csbi );
127       dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
128
129       //fill the entire screen with blanks
130       FillConsoleOutputCharacter( hConsole, (TCHAR)
 ' ', dwConSize, coordScreen, &cCharsWritten );
131       GetConsoleScreenBufferInfo( hConsole, &csbi );
132
133       //now set the buffer's attributes accordingly
134       FillConsoleOutputAttribute( hConsole,
 csbi.wAttributes,dwConSize, coordScreen, &cCharsWritten
 );
135
136       //put the cursor at (0, 0)
137       SetConsoleCursorPosition( hConsole, coord-
 Screen );
138
139       return;
140   }
141
142   /* FUNCTION: int RunDelivery(void)
143    *
144    * PURPOSE: This function executes the main deliv-
 ery executable loop.
145    *
146    * ARGUMENTS:   None
147    *
148    * RETURNS:     int ERR_CANNOT_OPEN_PIPE
 cannot open named pipe
149    *                 ERR_CANNOT_CREATE_THREAD    can-
 not create required threads
150    *                 ERR_SUCCESS                 suc-
 cessfull no error
151    *
152    *
153    * COMMENTS:    None
154    *
155    */
156
157   static int RunDelivery(void)
158   {
159       SECURITY_ATTRIBUTES sa;
160       int                 i;
161
162       cls();
163
164       PrintHeader();
165
166       printf("\n<Starting Delivery Service with %d
 Threads.>\n", iNumThreads);
167       printf("\nPress <Ctrl>C to exit.\n");
168
169       bDone = FALSE;
170       _beginthread( CheckKey, 0, NULL );
171
172       printf("\nWaiting for delivery pipe: ");
173
174       while( !bDone )
175       {
176           AnimateWait1();
```

```
177           if ( WaitNamedPipe(lpszPipeName,
 NMPWAIT_USE_DEFAULT_WAIT) )
178           {
179               sa.nLength              = sizeof(sa);
180               sa.lpSecurityDescriptor = NULL;
181               sa.bInheritHandle       = TRUE;
182
183               hPipe = CreateFile(lpszPipeName,
 GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ |
 FILE_SHARE_WRITE, NULL, OPEN_EXISTING,
184                   FILE_FLAG_OVERLAPPED, NULL);
185               if ( hPipe  == INVALID_HANDLE_VALUE )
186                   return ERR_CANNOT_OPEN_PIPE;
187               hComPort = CreateIoCompletion-
 Port(hPipe, NULL, 0, 256);
188               break;
189           }
190           Sleep(100);
191       }
192
193       if ( !bDone )
194       {
195          if ( _beginthread( DeliveryHandler, 0, NULL
 ) == -1 )
196               return ERR_CANNOT_CREATE_THREAD;
197
198           for(i=0; i<iNumThreads; i++)
199           {
200               if ( _beginthread( DeliveryThread, 0,
 NULL ) == -1 )
201                   return ERR_CANNOT_CREATE_THREAD;
202           }
203
204           printf(" \nRunning : ");
205
206           while( !bDone )
207               AnimateWait();
208       }
209
210       return ERR_SUCCESS;
211   }
212
213   /* FUNCTION: void AnimateWait1(void)
214    *
215    * PURPOSE: This function provides a visual indi-
 cator that the delivery executable is waiting for
216    *          the delivery pipe to appear.
217    *
218    * ARGUMENTS:   None
219    *
220    * RETURNS:     None
221    *
222    * COMMENTS:    None
223    *
224    */
225
226   static void AnimateWait1(void)
227   {
228       const static char szStr[] = "+-|*";
229       static char *ptr = (char *)szStr;
230
231       printf("%c\x8", *ptr);
232       ptr = (*(ptr+1)) ? ptr + 1 : (char *)szStr;
233       Sleep(100);
234
235       return;
236   }
237
238   /* FUNCTION: void AnimateWait(void)
239    *
240    * PURPOSE: This function provides a visual indi-
 cator that the delivery executable is waiting for
241    *          and processing transactions.
242    *
243    * ARGUMENTS:   None
244    *
245    * RETURNS:     None
246    *
247    * COMMENTS:    None
248    *
249    */
250
251   static void AnimateWait(void)
252   {
253       const static char szStr[] = "/-\\|/-\\|";
254       static char *ptr = (char *)szStr;
255
256       printf("%c\x8", *ptr);
257       ptr = (*(ptr+1)) ? ptr + 1 : (char *)szStr;
258       Sleep(100);
259
260       return;
261   }
262
263
264   /* FUNCTION: int Init(void)
265    *
266    * PURPOSE: This function prepares the delivery
 executable for processing.
267    *
268    * ARGUMENTS:   None
269    *
270    * RETURNS:     int iError      Error code if
 unsuccessfull
271    *                  ERR_SUCCESS No error success-
 full code
272    *
273    *
274    * COMMENTS:    None
275    *
276    */
277
278   static int Init(void)
279   {
280       int iError;
281
282       InitializeCriticalSection(&WriteLogCritical-
 Section);
283       InitializeCriticalSection(&DeliveryCritical-
 Section);
284
285       fpLog   = NULL;
286
287       if ( !(pDeliveryCache = mal-
 loc(sizeof(DELIVERY_PACKET) * iQSlotts)) )
288           return ERR_INSUFFICIENT_MEMORY;
289
290       memset(pDeliveryCache, 0,
 sizeof(DELIVERY_PACKET) * iQSlotts);
291
292       if ( (iError = ReadRegistrySettings()) )
293           return iError;
294
295       if ( (iError=OpenLogFile()) )
296           return iError;
297
298       //initialize db library for use
299   #ifdef USE_ODBC
300       if ( SQLAllocEnv(&henv) == SQL_ERROR )
301           return ERR_ODBC_SQLALLOCENV;
302   #else
303       dbinit();
304
305       // install Db Library error and message han-
 dlers
306       dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
307       dberrhandle((DBERRHANDLE_PROC)err_handler);
308   #endif
309
310       return ERR_SUCCESS;
311   }
312
313   /* FUNCTION: void Restore(void)
314    *
```

```
315     * PURPOSE: This function cleans up allocated
 objects to allow for termination of the
316     *          delivery executable.
317     *
318     * ARGUMENTS:   None
319     *
320     * RETURNS:     None
321     *
322     * COMMENTS:    None
323     *
324     */
325
326    static void Restore(void)
327    {
328        int iret, l, d;
329
330        DeleteCriticalSection(&WriteLogCriticalSec-
 tion);
331        DeleteCriticalSection(&DeliveryCriticalSec-
 tion);
332
333        l = 1;
334        iret = WriteFile(hPipe, &l, 1, &d, NULL);
335
336        if ( hPipe  != INVALID_HANDLE_VALUE )
337            iret = CloseHandle(hPipe);
338
339        if ( fpLog )
340            fclose(fpLog);
341
342        fpLog = NULL;
343
344    #ifdef USE_ODBC
345        SQLFreeEnv(henv);
346    #else
347        dbexit();
348    #endif
349
350        return;
351    }
352
353    /* FUNCTION: void ErrorMessage(int iError)
354     *
355     * PURPOSE: This function displays an error mes-
 sage in the delivery executable's console window.
356     *
357     * ARGUMENTS:   int     iError  error id to be dis-
 played
358     *
359     * RETURNS:     None
360     *
361     * COMMENTS:    None
362     *
363     */
364
365    static void ErrorMessage(int iError)
366    {
367        int i;
368
369        static SERRORMSG errorMsgs[] =
370        {
371            {   ERR_SUCCESS,
 "Success, no error."                          },
372            {   ERR_CANNOT_CREATE_THREAD,
 "Cannot create thread."                       },
373            {   ERR_DBGETDATA_FAILED,
 "Get data failed."                            },
374            {   ERR_REGISTRY_NOT_SETUP,
 "Registry not setup for tpcc."                },
375            {
 ERR_CANNOT_ACCESS_DELIVERY_FN,          "Cannot access
 ReadDelivery cache."             },
376            {   ERR_CANNOT_ACCESS_REGISTRY,
 "Cannot access registry key TPCC."            },
377            {
 ERR_CANNOT_CREATE_RESULTS_FILE,         "Cannot create
 results file."                   },
378            {   ERR_CANNOT_OPEN_PIPE,
 "Cannot open delivery pipe."                  },
379            {   ERR_READ_PIPE,
 "Reading Delivery Pipe."                      },
380            {   ERR_INSUFFICIENT_MEMORY,
 "Insufficient memory."                        },
381            {   ERR_ODBC_SQLALLOCENV,
 "Cannot allocated ODBC env handle."           },
382            {   ERR_SQL_ATTR_ODBC_VERSION,
 "Cannot set ODBC version."                    },
383             {
 ERR_SQL_ATTR_CONNECTION_POOLING,        "Cannot set
 Connection Pooling."             },
384        {  0,                  ""
 }
385        };
386
387        for(i=0; errorMsgs[i].szMsg[0]; i++)
388        {
389            if ( iError == errorMsgs[i].iError )
390            {
391                printf("\nError(%d): %s", iError,
 errorMsgs[i].szMsg);
392                if ( fpLog )
393                {
394                   EnterCriticalSection(&WriteLogCrit-
 icalSection);
395                    fprintf(fpLog, "*Error(%d):
 %s\r\n", iError, errorMsgs[i].szMsg);
396                    if ( bFlush )
397                        fflush(fpLog);
398                   LeaveCriticalSection(&WriteLogCrit-
 icalSection);
399                }
400                return;
401            }
402        }
403
404        printf("Error(%d): Unknown Error.");
405        EnterCriticalSection(&WriteLogCriticalSec-
 tion);
406        fprintf(fpLog, "*Error(%d): Unknown
 Error.\r\n", iError);
407        if ( bFlush )
408            fflush(fpLog);
409        LeaveCriticalSection(&WriteLogCriticalSec-
 tion);
410
411        return;
412    }
413
414    /* FUNCTION: BOOL GetParameters(int argc, char
 *argv[])
415     *
416     * PURPOSE: This function parses the command line
 passed in to the delivery executable, initializing
417     *          and filling in global variable parame-
 ters.
418     *
419     * ARGUMENTS:   int     argc    number of command
 line arguments passed to delivery
420     *              char    *argv[] array of command
 line argument pointers
421     *
422     * RETURNS:     BOOL    FALSE   parameter read suc-
 cessfull
423     *                      TRUE    user has requested
 parameter information screen be displayed.
424     *
425     * COMMENTS:    None
426     *
427     */
428
429    static BOOL GetParameters(int argc, char *argv[])
430    {
431        int i;
432
```

```
433
434         szServer[0]     = 0;
435         szPassword[0]   = 0;
436         bFlush          = FALSE;
437         strcpy(szDatabase, "tpcc");
438         strcpy(szUser, "sa");
439
440         for(i=0; i<argc; i++)
441         {
442            if ( argv[i][0] == '-' || argv[i][0] == '/'
 )
443             {
444                 switch(argv[i][1])
445                 {
446                     case 'S':
447                     case 's':
448                         strcpy(szServer, argv[i]+2);
449                         break;
450                     case 'D':
451                     case 'd':
452                         strcpy(szDatabase, argv[i]+2);
453                         break;
454                     case 'U':
455                     case 'u':
456                         strcpy(szUser, argv[i]+2);
457                         break;
458                     case 'P':
459                     case 'p':
460                         strcpy(szPassword, argv[i]+2);
461                         break;
462                     case 'F':
463                     case 'f':
464                         bFlush = TRUE;  //turn on
 delilog flush when written.
465                         break;
466                     case '?':
467                         return TRUE;
468                 }
469             }
470         }
471         return FALSE;
472     }
473
474     /* FUNCTION: void PrintParameters(void)
475      *
476      * PURPOSE: This function displays the supported
 command line flags.
477      *
478      * ARGUMENTS:   None
479      *
480      * RETURNS:     None
481      *
482      * COMMENTS:    None
483      *
484      */
485
486     static void PrintParameters(void)
487     {
488         PrintHeader();
489         printf("DELISRV:\n\n");
490         printf("Parame-
 ter
 Default\n");
491         printf("-------------------------------------
 ---------------------------------\n");
492         printf("-S
 Server
              \n");
493         printf("-D Data-
 base
 tpcc   \n");
494         printf("-U User-
 name                                          sa
 \n");
495         printf("-P Pass-
 word
           \n");
496         printf("-F Flush output to delilog file when
 written.                    OFF    \n");
497         printf("-? This help screen\n\n");
498         printf("Note:  Command line switches are NOT
 case sensitive.\n");
499
500         return;
501     }
502
503     /* FUNCTION: void PrintHeader(void)
504      *
505      * PURPOSE: This function displays the delivery
 executable's banner information.
506      *
507      * ARGUMENTS:   None
508      *
509      * RETURNS:     None
510      *
511      * COMMENTS:    None
512      *
513      */
514
515     static void PrintHeader(void)
516     {
517
 printf("*********************************************
 ***\n");
518
 printf("*
 *\n");
519     #ifdef USE_ODBC
520         printf("*  Microsoft SQL Server 6.5
 (ODBC)              *\n");
521     #else
522         printf("*  Microsoft SQL Server 6.5
 (DBLIB)             *\n");
523     #endif
524
 printf("*
 *\n");
525         printf("*  HTML TPC-C BENCHMARK KIT: Delivery
 Server    *\n");
526         printf("*  Version
 %d.%2.2d.%3.3d                              *\n", ver-
 sionMS, versionMM, versionLS);
527
 printf("*
 *\n");
528
 printf("*********************************************
 ***\n\n");
529
530         return;
531     }
532
533     /* FUNCTION: int ReadRegistrySettings(void)
534      *
535      * PURPOSE: This function reads the system regis-
 try filling in required key parameters.
536      *
537      * ARGUMENTS:   None
538      *
539      * RETURNS:     int ERR_REGISTRY_NOT_SETUP
 registry not setup tpcc.exe needs to be run
540      *                                                to
 setup registry.
541      *                   ERR_SUCCESS
 Reqistry read Successfull, no error
542      *
543      *
544      * COMMENTS:    None
545      */
546
547     static int ReadRegistrySettings(void)
548     {
549         HKEY    hKey;
550         DWORD   size;
```

```
551        DWORD   type;
552        char    szTmp[256];
553
554        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFT-
 WARE\\Microsoft\\TPCC", 0, KEY_READ, &hKey) !=
 ERROR_SUCCESS )
555            return ERR_REGISTRY_NOT_SETUP;
556
557        size = sizeof(szTmp);
558
559        iNumThreads = 4;
560        if ( RegQueryValueEx(hKey, "NumberOfDelivery-
 Threads", 0, &type, szTmp, &size) == ERROR_SUCCESS )
561            iNumThreads = atoi(szTmp);
562            if ( !iNumThreads )
563                iNumThreads = 4;
564
565        iDelayMs = 1000;
566        if ( RegQueryValueEx(hKey, "BackoffDelay", 0,
 &type, szTmp, &size) == ERROR_SUCCESS )
567            iDelayMs = atoi(szTmp);
568            if ( !iDelayMs )
569                iDelayMs = 1000;
570
571        iDeadlockRetry = 3;
572       if ( RegQueryValueEx(hKey, "DeadlockRetry", 0,
 &type, szTmp, &size) == ERROR_SUCCESS )
573            iDeadlockRetry = atoi(szTmp);
574        if ( !iDeadlockRetry )
575            iDeadlockRetry = 3;
576
577        RegCloseKey(hKey);
578
579        return ERR_SUCCESS;
580    }
581
582    /* FUNCTION: void CheckKey(void *ptr)
583     *
584     * PURPOSE: This function checks for a key press
 on the delivery executable's console. If the
585     *          key press is a Ctrl C then the execu-
 tion termination flag variable bDone is set to
586     *          TRUE which will start the termination
 of the delivery executable.
587     *
588     * ARGUMENTS:   void    *ptr    dummy argument
 passed in though thread manager, unused NULL.
589     *
590     * RETURNS:     None
591     *
592     * COMMENTS:    None
593     *
594     */
595
596    static void CheckKey(void *ptr)
597    {
598        while( _getch() != CTRL_C)
599            ;
600        bDone = TRUE;
601
602        return;
603    }
604
605    /* FUNCTION: void DeliveryHandler( void *ptr )
606     *
607     * PURPOSE: This function is executed in it's own
 thread what it does is to check for delivery
608     *          postings in the delivery named pipe. If
 any are present then it pulls them off and
609     *          places them in the next available
 delivery queue array element.
610     *
611     * ARGUMENTS:   void    *ptr    dummy argument
 passed in though thread manager, unused NULL.
612     *
613     * RETURNS:     None
614     *
615     * COMMENTS:    None
616     *
617     */
618
619    static void DeliveryHandler( void *ptr )
620    {
621        int i;
622        int size;
623        int iError;
624
625        while( !bDone )
626        {
627            for(i=0; i<iQSlotts; i++)
628            {
629                if ( !pDeliveryCache[i].bInUse )
630                    break;
631            }
632            if ( i < iQSlotts )
633            {
634                EnterCriticalSection(&DeliveryCriti-
 calSection);
635                pDeliveryCache[i].bInUse = TRUE;
636                LeaveCriticalSection(&DeliveryCriti-
 calSection);
637            }
638            else
639            {
640                EnterCriticalSection(&DeliveryCriti-
 calSection);
641                if ( !(pDeliveryCache =
 (LPDELIVERY_PACKET)realloc(pDeliveryCache,
 sizeof(DELIVERY_PACKET) * (iQSlotts+512))) )
642                {
643                    ErrorMes-
 sage(ERR_INSUFFICIENT_MEMORY);
644                   LeaveCriticalSection(&DeliveryCrit-
 icalSection);
645                    return;
646                }
647                for(i=iQSlotts; i<iQSlotts+512; i++)
648                    pDeliveryCache[i].bInUse = FALSE;
649                i = iQSlotts;
650                pDeliveryCache[i].bInUse = TRUE;
651                LeaveCriticalSection(&DeliveryCriti-
 calSection);
652            }
653
654            pDeliveryCache[i].ov.Offset         = i;
655            pDeliveryCache[i].ov.Internal       = 0;
656            pDeliveryCache[i].ov.InternalHigh   = 0;
657            pDeliveryCache[i].ov.OffsetHigh     = 1;
658           pDeliveryCache[i].ov.hEvent        = NULL;
659
660            while( !bDone )
661            {
662                if ( ReadFile(hPipe, &pDelivery-
 Cache[i].trans, sizeof(DELIVERY_TRANSACTION), &size,
 &pDeliveryCache[i].ov) )
663                    break;
664                if ( bDone )
665                    break;
666                iError = GetLastError();
667                if ( iError == ERROR_IO_PENDING )
668                {
669                    while( pDeliveryCache[i].ov.Off-
 setHigh )
670                        Sleep(10);
671                    break;
672                }
673                else
674                {
675                    ErrorMessage(ERR_READ_PIPE);
676                    return;
677                }
678            }
679            Sleep(1);
680        }
```

```
681
682         return;
683     }
684
685     /* FUNCTION: void DeliveryThread( void *ptr )
686      *
687      * PURPOSE: This function is executed inside the
 delivery threads. The queue array
688      *          is continuously check and if any array
 elements are in use then the
689      *          array entry is read, cleared and this
 function processes it.
690      *
691      * ARGUMENTS:   void    *ptr    dummy argument
 passed in though thread manager, unused NULL.
692      *
693      * RETURNS:     None
694      *
695      * COMMENTS:    The registry key
 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
696      *              value NumberOfDeliveryThreads con-
 trols how many of these
697      *              functions are running. The
 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
698      *              value BackoffDelay controls the
 amount of time this function waits
699      *              between checks of the delivery
 queue.
700      *
701      */
702
703     static void DeliveryThread( void *ptr )
704     {
705         int             size;
706         int             key;
707         LPOVERLAPPED    pov;
708         DELIVERY        delivery;
709         int             iError;
710
711         if ( SQLOpenConnection(&delivery.dbproc,
 szServer, szDatabase, szUser, szPassword, &deliv-
 ery.spid) )
712             return; //error posting tbd
713
714         //while delisrv running i.e. user has not
 requested termination
715         while( !bDone )
716         {
717             if ( GetQueuedCompletionStatus(hComPort,
 &size, &key, &pov, (DWORD)-1) )
718             {
719                 pov->OffsetHigh = 0;    //clear to
 notify delivery handler ok to read another entry.
720                 //some delivery to do so process it
721                 memcpy(&delivery.queue, &pDelivery-
 Cache[pov->Offset].trans.queue, sizeof(SYSTEMTIME));
722                 delivery.w_id           = pDelivery-
 Cache[pov->Offset].trans.w_id;
723                 delivery.o_carrier_id   = pDelivery-
 Cache[pov->Offset].trans.o_carrier_id;
724
725                 if ( (iError=SQLDelivery(&delivery)) )
726                 {
727                     ErrorMessage(iError);
728                     printf("Running : ");
729                     continue;
730                 }
731
732                 //update log
733                 WriteLog(&delivery);
734
735                 EnterCriticalSection(&DeliveryCriti-
 calSection);
736                 pDeliveryCache[pov->Offset].bInUse =
 FALSE;
737                 LeaveCriticalSection(&DeliveryCriti-
 calSection);
738             }
739         }
740
741         return;
742     }
743
744     /* FUNCTION: static int err_handler(DBPROCESS
 *dbproc, int severity, int dberr, int oserr, char
 *dberrstr, char *oserrstr)
745      *
746      * PURPOSE: This function handles DB-Library
 errors
747      *
748      * ARGUMENTS:   DBPROCESS       *dbproc
 DBPROCESS id pointer
749      *              int             severity
 severity of error
750      *             int            dberr          error
 id
751      *             int            oserr          oper-
 ating system specific error code
752      *              char            *dberrstr
 printable error description of dberr
753      *              char            *oserrstr
 printable error description of oserr
754      *
755      * RETURNS:     int            INT_CONTINUE    con-
 tinue if error is SQLETIME else INT_CANCEL action
756      *
757      * COMMENTS:    None
758      *
759      */
760
761     #ifndef USE_ODBC
762     static int err_handler(DBPROCESS *dbproc, int
 severity, int dberr, int oserr, char *dberrstr, char
 *oserrstr)
763     {
764         if (oserr != DBNOERR)
765             printf("(%d) %s", oserr, oserrstr);
766
767         if ((dbproc == NULL) || (DBDEAD(dbproc)))
768             ExitThread((unsigned long)-1);
769
770         return INT_CONTINUE;
771     }
772     #endif
773
774     /* FUNCTION: static int msg_handler(DBPROCESS
 *dbproc, DBINT msgno, int msgstate, int severity, char
 *msgtext)
775      *
776      * PURPOSE: This function handles DB-Library SQL
 Server error messages
777      *
778      * ARGUMENTS:   DBPROCESS       *dbproc
 DBPROCESS id pointer
779      *             DBINT          msgno           mes-
 sage number
780      *             int            msgstate        mes-
 sage state
781      *             int            severity        mes-
 sage severity
782      *              char            *msgtext
 printable message description
783      *
784      * RETURNS:     int            INT_CONTINUE    con-
 tinue if error is SQLETIME else INT_CANCEL action
785      *                             INT_CANCEL      can-
 cel operation
786      *
787      * COMMENTS:    This function also sets the dead
 lock dbproc variable if necessary.
788      *
789      */
790     static 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
            if (dbgetuserdata(dbproc) != NULL)
                *((BOOL *) dbgetuserdata(dbproc)) =
 TRUE;
            else
              printf("\nError, dbgetuserdata returned
 NULL.\n");

            return INT_CONTINUE;

        }

        if (msgno == 0)
            return INT_CONTINUE;
        else
            printf("SQL Server Message (%ld) : %s\n",
 msgno, msgtext);
        return INT_CANCEL;
    }

    /* FUNCTION: BOOL SQLOpenConnection(DBPROCESS
 **dbproc, char *server, char *database, char *user,
 char *password, int *spid)
     *
     * PURPOSE: This function opens the sql connection
 for use.
     *
     * ARGUMENTS:   DBPROCESS      **dbproc    pointer
 to returned DBPROCESS
     *              char            *server     SQL
 server name
     *              char            *database   SQL
 server database
     *             char           *user       user name
     *              char            *password   user
 password
     *             int             *spid       pointer
 to returned spid
     *
     * RETURNS:     BOOL    FALSE   if successfull
     *                      TRUE    if an error occurs
     *
     * COMMENTS:    None
     *
     */

    #ifdef USE_ODBC
        static BOOL SQLOpenConnection(DBPROCESS
 **dbproc, char *server, char *database, char *user,
 char *password, int *spid)
        {

            RETCODE     rc;
            char        buffer[30];

            *dbproc = (DBPROCESS *)mal-
 loc(sizeof(DBPROCESS));
            if ( !*dbproc )
                return TRUE;

            //set pECB data into dbproc
            dbsetuserdata(*dbproc, mal-
 loc(sizeof(BOOL)));
            *((BOOL *)dbgetuserdata(*dbproc)) = FALSE;

            if ( SQLAllocConnect(henv, &(*dbproc)-
 >hdbc) == SQL_ERROR )
                return TRUE;

            if ( SQLSetConnectOption((*dbproc)->hdbc,
 SQL_PACKET_SIZE, 4096) == SQL_ERROR )
                return TRUE;

            rc = SQLConnect((*dbproc)->hdbc, server,
 SQL_NTS, user, SQL_NTS, password, SQL_NTS);
            if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
                return TRUE;
            rc = SQLAllocStmt((*dbproc)->hdbc,
 &(*dbproc)->hstmt);
            if (rc == SQL_ERROR)
                return TRUE;

            strcpy(buffer, "use tpcc");

            rc = SQLExecDirect((*dbproc)->hstmt,
 buffer, SQL_NTS);
            if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
                return TRUE;

            SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);
            sprintf(buffer,"set nocount on");
            rc = SQLExecDirect((*dbproc)->hstmt,
 buffer, SQL_NTS);
            if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
                return TRUE;
            SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);

            sprintf(buffer,"select @@spid");

            rc = SQLExecDirect((*dbproc)->hstmt,
 buffer, SQL_NTS);
            if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
                return TRUE;

            if ( SQLBindCol((*dbproc)->hstmt, 1,
 SQL_C_SSHORT, &(*dbproc)->spid, 0, NULL) == SQL_ERROR )
                return TRUE;

            if ( SQLFetch((*dbproc)->hstmt) ==
 SQL_ERROR )
                return TRUE;

            SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);

            return FALSE;
        }
    #else
        static BOOL SQLOpenConnection(DBPROCESS
 **dbproc, char *server, char *database, char *user,
 char *password, int *spid)
        {
            LOGINREC   *login;

            login = dblogin();
            DBSETLUSER(login, user);
            DBSETLPWD(login,  password);

            DBSETLPACKET(login, (USHORT)DEFCLPACK-
 SIZE);

            if ((*dbproc = dbopen(login, server )) ==
 NULL)
                return TRUE;

            // Use the the right database
            dbuse(*dbproc, database);

            dbsetuserdata(*dbproc, mal-
 loc(sizeof(BOOL)));
            *((BOOL *)dbgetuserdata(*dbproc)) = FALSE;

            dbcmd(*dbproc, "select @@spid");
```

```
911
912             dbsqlexec(*dbproc);
913             while (dbresults(*dbproc) !=
 NO_MORE_RESULTS)
914             {
915                 dbbind(*dbproc, 1, SMALLBIND, (DBINT)
 0, (BYTE *) spid);
916                 while (dbnextrow(*dbproc) !=
 NO_MORE_ROWS);
917             }
918             dbcmd(*dbproc, "set nocount on");
919
920             dbsqlexec(*dbproc);
921             while (dbresults(*dbproc) !=
 NO_MORE_RESULTS)
922                 while (dbnextrow(*dbproc) !=
 NO_MORE_ROWS);
923
924             return FALSE;
925         }
926     #endif
927
928     //queue time, end time, elapsed time, w_id,
 o_carrier_id, o_id1, ... o_id10
929     /* FUNCTION: void WriteLog(LPDELIVERY pDelivery)
930      *
931      * PURPOSE: This function writes the delivery
 results to the delivery log file.
932      *
933      * ARGUMENTS:   LPDELIVERY  pDelivery   Pointer to
 delivery information.
934      *
935      * RETURNS:     None
936      *
937      * COMMENTS:    None
938      *
939      */
940
941     static void WriteLog(LPDELIVERY pDelivery)
942     {
943         int elapsed;
944
945         CalculateElapsedTime(&elapsed, &pDelivery-
 >queue, &pDelivery->trans_end);
946
947         EnterCriticalSection(&WriteLogCriticalSec-
 tion);
948
949         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\r\
 n",
950             pDelivery->trans_end.wYear - 1900, pDeliv-
 ery->trans_end.wMonth, pDelivery->trans_end.wDay,
951             pDelivery->queue.wHour, pDelivery-
 >queue.wMinute, pDelivery->queue.wSecond, pDelivery-
 >queue.wMilliseconds,
952             pDelivery->trans_end.wHour, pDelivery-
 >trans_end.wMinute, pDelivery->trans_end.wSecond, pDe-
 livery->trans_end.wMilliseconds,
953             elapsed,
954             pDelivery->w_id, pDelivery->o_carrier_id,
955             pDelivery->o_id[0], pDelivery->o_id[1],
 pDelivery->o_id[2], pDelivery->o_id[3],
956             pDelivery->o_id[4], pDelivery->o_id[5],
 pDelivery->o_id[6], pDelivery->o_id[7],
957             pDelivery->o_id[8], pDelivery->o_id[9] );
958
959         if ( bFlush )
960             fflush(fpLog);
961
962         LeaveCriticalSection(&WriteLogCriticalSec-
 tion);
963
964         return;
965     }
966
967     /* FUNCTION: void CalculateElapsedTime(int *pEl-
 apsed, LPSYSTEMTIME lpBegin, LPSYSTEMTIME lpEnd)
968      *
969      * PURPOSE: This function calculates the elapsed
 time a delivery transaction took.
970      *
971      * ARGUMENTS:   int            *pElapsed   pointer
 to int variable to receive calculated elapsed
972      *                                          time in
 milliseconds.
973      *             LPSYSTEMTIME    lpBegin     Pointer
 to system time structure containing
974      *                                            trans-
 action beginning time.
975      *             LPSYSTEMTIME    lpEnd       Pointer
 to system time structure containing
976      *                                            trans-
 action ending time.
977      * RETURNS:     None
978      *
979      * COMMENTS:    None
980      *
981      */
982
983     static void CalculateElapsedTime(int *pElapsed,
 LPSYSTEMTIME lpBegin, LPSYSTEMTIME lpEnd)
984     {
985         int         beginSeconds;
986         int         endSeconds;
987
988
989         beginSeconds = (lpBegin->wHour * 3600000) +
 (lpBegin->wMinute * 60000) + (lpBegin->wSecond * 1000)
 + lpBegin->wMilliseconds;
990         endSeconds = (lpEnd->wHour * 3600000) +
 (lpEnd->wMinute * 60000) + (lpEnd->wSecond * 1000) +
 lpEnd->wMilliseconds;
991         *pElapsed = endSeconds - beginSeconds;
992
993         //check for day boundry, this will function
 for 24 hour period however it will not work over 48
 hours.
994         if ( *pElapsed < 0 )
995             *pElapsed = *pElapsed + (24 * 60 * 60 *
 1000);
996
997         return;
998     }
999
1000    /* FUNCTION: int SQLDelivery(DELIVERY *pDelivery)
1001     *
1002     * PURPOSE: This function processes the delivery
 transaction.
1003     *
1004     * ARGUMENTS:   DELIVERY        *pDelivery
 Pointer to delivery transaction structure
1005     *
1006     * RETURNS:     int ERR_DBGETDATA_FAILED
 Delivery get data operation failed.
1007     *                 ERR_SUCCESS
 Delivery successfull, no error
1008     *
1009     *
1010     * COMMENTS:    None
1011     *
1012     */
1013
1014    #ifdef USE_ODBC
1015         static int SQLDelivery(DELIVERY *pDelivery)
1016         {
1017             int     i;
1018             int     deadlock_count;
1019             SDWORD  iLength[10];
1020             BOOL    bDeadlock;
1021
1022             deadlock_count = 0;
1023
```

```
1024            // Start new delivery
1025            while ( TRUE )
1026            {
1027                BindParameter(pDelivery->dbproc, 1,
  SQL_C_SSHORT, SQL_SMALLINT, 0, 0, &pDelivery->w_id, 0);
1028                BindParameter(pDelivery->dbproc, 2,
  SQL_C_SSHORT, SQL_SMALLINT, 0, 0, &pDelivery-
  >o_carrier_id, 0);
1029
1030                if ( ExecuteStatement(pDelivery-
  >dbproc, "{call tpcc_delivery (?, ?)}") )
1031                    return 1;
1032                bDeadlock = *((BOOL *)dbgetuser-
  data(pDelivery->dbproc));
1033                if ( !bDeadlock )
1034                {
1035                    for (i=0;i<10;i++)
1036                    {
1037                        if ( BindColumn(pDelivery-
  >dbproc, (UWORD)(i+1), SQL_C_SLONG, &pDelivery-
  >o_id[i], 0, &iLength[i]) )
1038                            return 1;
1039                    }
1040                  if ( GetResults(pDelivery->dbproc)
  )
1041                        return 1;
1042                    for(i=0; i<10; i++)
1043                    {
1044                        if ( iLength[i] <= 0 )
1045                            pDelivery->o_id[i] = 0;
1046                    }
1047                }
1048               SQLFreeStmt(pDelivery->dbproc->hstmt,
  SQL_CLOSE);
1049                if ( !SQLDetectDeadlock(pDelivery-
  >dbproc) )
1050                    break;
1051                deadlock_count++;
1052                Sleep(10 * deadlock_count);
1053            }
1054            GetLocalTime(&pDelivery->trans_end);
1055
1056            return ERR_SUCCESS;
1057        }
1058    #else
1059        static int SQLDelivery(DELIVERY *pDelivery)
1060        {
1061            RETCODE rc;
1062            int     i;
1063            int     deadlock_count;
1064            BYTE    *pData;
1065
1066            deadlock_count = 0;
1067
1068
1069            // Start new delivery
1070            while ( TRUE )
1071            {
1072                if (dbrpcinit(pDelivery->dbproc,
  "tpcc_delivery", 0) == SUCCEED)
1073                {
1074                  dbrpcparam(pDelivery->dbproc, NULL,
  0, SQLINT2, -1, -1, (BYTE *)&pDelivery->w_id);
1075                  dbrpcparam(pDelivery->dbproc, NULL,
  0, SQLINT1, -1, -1, (BYTE *) &pDelivery->o_carrier_id);
1076
1077                  if (dbrpcexec(pDelivery->dbproc) ==
  SUCCEED)
1078                    {
1079                      while (((rc = dbresults(pDeliv-
  ery->dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
1080                        {
1081                          while (((rc = dbnextrow(pDe-
  livery->dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
1082                            {
1083                                for (i=0;i<10;i++)
1084                                {
1085                                    if(pData=dbdata(pDe-
  livery->dbproc, i+1))
1086                                        pDelivery-
  >o_id[i] = *((DBINT *)pData);
1087                                    else
1088                                        pDelivery-
  >o_id[i] = 0;
1089                                }
1090                            }
1091                        }
1092                    }
1093                }
1094                if ( !SQLDetectDeadlock(pDelivery-
  >dbproc) )
1095                    break;
1096                deadlock_count++;
1097                Sleep(10 * deadlock_count);
1098            }
1099            GetLocalTime(&pDelivery->trans_end);
1100
1101            return ERR_SUCCESS;
1102        }
1103    #endif
1104
1105    /* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS
  *dbproc)
1106     *
1107     * PURPOSE: This function is used to check for
  deadlock conditions.
1108     *
1109     * ARGUMENTS:   DBPROCESS       *dbproc     DBPRO-
  CESS to check
1110     *
1111     * RETURNS:     BOOL    FALSE              No lock
  condition present
1112     *                      TRUE                Lock
  condition detected
1113     *
1114     * COMMENTS:    None
1115     *
1116     */
1117
1118    static BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
1119    {
1120        if (*((BOOL *) dbgetuserdata(dbproc)) == TRUE)
1121        {
1122            *((BOOL *) dbgetuserdata(dbproc)) = FALSE;
1123            return TRUE;
1124        }
1125        return FALSE;
1126    }
1127
1128    /* FUNCTION: int OpenLogFile(void)
1129     *
1130     * PURPOSE: This function opens the delivery log
  file for use.
1131     *
1132     * ARGUMENTS:   None
1133     *
1134     * RETURNS:     int
  ERR_REGISTRY_NOT_SETUP          Registry not setup.
1135     *                  ERR_CANNOT_CREATE_RESULTS_FILE
  Cannot create results log file.
1136     *                   ERR_SUCCESS
  Log file successfully opened
1137     *
1138     *
1139     * COMMENTS:    None
1140     *
1141     */
1142
1143    static int OpenLogFile(void)
1144    {
1145        HKEY    hKey;
1146        BOOL    bRc;
1147        BYTE    szTmp[256];
1148        char    szKey[256];
```

```
1149        char    szLogPath[256];
1150        DWORD   size;
1151        DWORD   sv;
1152        int     len;
1153        char    *ptr;
1154
1155        szLogPath[0] =  0;
1156        bRc = TRUE;
1157        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SYS-
  TEM\\CurrentControlSet\\Services\\W3SVC\\Parame-
  ters\\Virtual Roots", 0, KEY_ALL_ACCESS, &hKey) ==
  ERROR_SUCCESS )
1158        {
1159            sv = sizeof(szKey);
1160            size = sizeof(szTmp);
1161
1162            if ( RegEnumValue(hKey, 0, szKey, &sv,
  NULL, NULL, szTmp, &size) == ERROR_SUCCESS )
1163            {
1164                strcpy(szLogPath, szTmp);
1165                bRc = FALSE;
1166            }
1167            RegCloseKey(hKey);
1168        }
1169
1170        if ( bRc )
1171            return ERR_REGISTRY_NOT_SETUP;
1172
1173        if ( (ptr = strchr(szLogPath, ',')) )
1174            *ptr = 0;
1175
1176        len = strlen(szLogPath);
1177        if ( szLogPath[len-1] != '\\' )
1178        {
1179            szLogPath[len] = '\\';
1180            szLogPath[len+1] = 0;
1181        }
1182        strcat(szLogPath, "delilog.");
1183
1184        fpLog = fopen(szLogPath, "ab");
1185
1186        if ( !fpLog )
1187            return ERR_CANNOT_CREATE_RESULTS_FILE;
1188
1189        return ERR_SUCCESS;
1190    }
1191
1192    #ifdef USE_ODBC
1193
1194        /* FUNCTION: void dbsetuserdata(PDBPROCESS
  dbproc, void *uPtr)
1195         *
1196         * PURPOSE: This function sets a user pointer
  in a dbproc structure
1197         *          This functionality is not provided
  in odbc so this function
1198         *          provides it.
1199         *
1200         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1201         *              void        *uPtr   returned
  data user pointer
1202         *
1203         * RETURNS:     none
1204         *
1205         * COMMENTS:    The caller is responsible for
  the contents of the uPtr.
1206         *
1207         */
1208
1209        void dbsetuserdata(PDBPROCESS dbproc, void
  *uPtr)
1210        {
1211            dbproc->uPtr = uPtr;
1212        }
1213
1214        /* FUNCTION: void dbsetuserdata(PDBPROCESS
  dbproc, void *uPtr)
1215         *
1216         * PURPOSE: This function returns the user
  pointer stored in a dbproc structure
1217         *          This functionality is not provided
  in odbc so this function
1218         *          provides it.
1219         *
1220         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1221         *
1222         * RETURNS:     none
1223         *
1224         * COMMENTS:    The returned pointer is placed
  in the dbproc structure by the dbsetuserdata() API.
1225         *
1226         */
1227
1228        void *dbgetuserdata(PDBPROCESS dbproc)
1229        {
1230            return dbproc->uPtr;
1231        }
1232
1233        /* FUNCTION: void BindParameter(PDBPROCESS
  dbproc, UWORD ipar, SWORD fCType, SWORD fSqlType,
  UDWORD cbColDef, SWORD ibScale, PTR rgbValue, SDWORD
  cbValueMax)
1234         *
1235         * PURPOSE: This function wraps the function-
  ality provided by the SQLBindParameter
1236         *          allowing error process so that each
  bind call does not need to provide
1237         *          error and message checking.
1238         *
1239         * ARGUMENTS:   PDBPROCESS  dbproc     pointer
  to odbc dbprocess structure
1240         *              UWORD   ipar        Parameter
  number, ordered sequentially left to right, starting at
  1.
1241         *             SWORD   fParamType  The type of
  the parameter.
1242         *              SWORD   fCType      The C data
  type of the parameter.
1243         *             SWORD   fSqlType    The SQL data
  type of the parameter.
1244         *              UDWORD  cbColDef    The preci-
  sion of the column or expression
1245         *                                  of the
  corresponding parameter marker.
1246         *              SWORD   ibScale     The scale
  of the column or expression of the corresponding
1247         *                                  parameter
  marker.
1248         *              PTR     rgbValue    A pointer
  to a buffer for the parameters data.
1249         *              SDWORD  cbValueMax  Maximum
  length of the rgbValue buffer.
1250         *              void        *uPtr   returned
  data user pointer
1251         *
1252         * RETURNS:     none
1253         *
1254         * COMMENTS:    The returned pointer is placed
  in the dbproc structure by the dbset
1255         *
1256         */
1257
1258        void BindParameter(PDBPROCESS dbproc, UWORD
  ipar, SWORD fCType, SWORD fSqlType, UDWORD cbColDef,
  SWORD ibScale, PTR rgbValue, SDWORD cbValueMax)
1259        {
1260            RETCODE rc;
1261
1262           rc = SQLBindParameter(dbproc->hstmt, ipar,
  SQL_PARAM_INPUT, fCType, fSqlType, cbColDef, ibScale,
  rgbValue, cbValueMax, NULL);
1263            if (rc == SQL_ERROR)
```

```
1264                ODBCError(dbproc);
1265            return;
1266        }
1267
1268        /* FUNCTION: void ODBCError(PDBPROCESS dbproc)
1269         *
1270         * PURPOSE: This function wraps the odbc error
  call so that the dblib msg_handler is called.
1271         *          This allows the deadlock flag in
  the dbproc user data structure pEcbInfo in
1272         *          dbproc to be set if necessary.
1273         *
1274         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1275         *
1276         * RETURNS:     none
1277         *
1278         * COMMENTS:    none
1279         *
1280         */
1281
1282        void ODBCError(PDBPROCESS dbproc)
1283        {
1284            SDWORD      lNativeError;
1285            char        szState[6];
1286            char        szMsg[SQL_MAX_MESSAGE_LENGTH];
1287
1288            while( SQLError(henv, dbproc->hdbc,
  dbproc->hstmt, szState, &lNativeError, szMsg,
  sizeof(szMsg), NULL) == SQL_SUCCESS )
1289            {
1290              msg_handler(dbproc, lNativeError, 0, 0,
  szMsg);
1291                if ( !lNativeError )
1292                {
1293                    printf("\nODBC Error State = %s,
  %s\n", szState, szMsg);
1294                    printf("Running : ");
1295                }
1296            }
1297
1298            return;
1299        }
1300
1301        /* FUNCTION: BOOL ExecuteStatement(PDBPROCESS
  dbproc, szStatement)
1302         *
1303         * PURPOSE: This function wraps the odbc
  SQLExecDirect API so that error handling and
1304         *          and deadlock are taken care of in
  a common location.
1305         *
1306         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1307         *             char       *szStatement    sql
  stored procedure statement to be executed.
1308         *
1309         * RETURNS:     none
1310         *
1311         * COMMENTS:    none
1312         *
1313         */
1314
1315        BOOL ExecuteStatement(PDBPROCESS dbproc, char
  *szStatement)
1316        {
1317            RETCODE     rc;
1318
1319            rc = SQLExecDirect(dbproc->hstmt, szState-
  ment, SQL_NTS);
1320            if (rc != SQL_SUCCESS && rc !=
  SQL_SUCCESS_WITH_INFO)
1321            {
1322                ODBCError(dbproc);
1323               if ( *((BOOL *)dbgetuserdata(dbproc)) )
1324                    return FALSE;
1325                return TRUE;
1326            }
1327            return FALSE;
1328        }
1329
1330        /* FUNCTION: BOOL BindColumn(PDBPROCESS
  dbproc, SQLUSMALLINT icol, SQLSMALLINT fCType,
  SQLPOINTER rgbValue, SQLINTEGER cbValueMax, SDWORD FAR
  *piLength)
1331         *
1332         * PURPOSE: This function wraps the odbc SQL-
  BindCol API so that error handling and
1333         *          and deadlock are taken care of in
  a common location.
1334         *
1335         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1336         *             UWORD   icol            Column
  number of result data, ordered sequentially left to
  right, starting at 1.
1337         *              SWORD   fCType          The C
  data type of the result data. SQL_C_BINARY, SQL_C_BIT,
  SQL_C_BOOKMARK,
1338         *
  SQL_C_CHAR, SQL_C_DATE, SQL_C_DEFAULT, SQL_C_DOUBLE,
  SQL_C_FLOAT, SQL_C_SLONG,
1339         *
  SQL_C_SSHORT, SQL_C_STINYINT, SQL_C_TIME,
  SQL_C_TIMESTAMP, SQL_C_ULONG,
1340         *
  SQL_C_USHORT, SQL_C_UTINYINT, SQL_C_DEFAULT
1341         *             PTR     rgbValue        Pointer
  to storage for the data. If rgbValue is a null pointer,
  the
1342         *                                     driver
  unbinds the column.
1343         *             SDWORD  cbValueMax      Maximum
  length of the rgbValue buffer. For character data, rgb-
  Value
1344         *                                      must
  also include space for the null-termination byte.
1345         *             SDWORD *piLength        Pointer
  to variable to receive length of returned data.
1346         * RETURNS:     none
1347         *
1348         * COMMENTS:    none
1349         *
1350         */
1351
1352        BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALL-
  INT icol, SQLSMALLINT fCType, SQLPOINTER rgbValue,
  SQLINTEGER cbValueMax, SDWORD *piLength)
1353        {
1354            RETCODE     rc;
1355
1356            rc = SQLBindCol(dbproc->hstmt, icol,
  fCType, rgbValue, cbValueMax, piLength);
1357            if ( rc == SQL_ERROR )
1358            {
1359                ODBCError(dbproc);
1360                return TRUE;
1361            }
1362            return FALSE;
1363        }
1364
1365        /* FUNCTION: BOOL GetResults(PDBPROCESS
  dbproc)
1366         *
1367         * PURPOSE: This function wraps the odbc
  SQLFetch API so that error handling and
1368         *          and deadlock are taken care of in
  a common location.
1369         *
1370         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1371         *
1372         * RETURNS:     none
1373         *
```

```
1374         * COMMENTS:    none
1375         *
1376         */
1377
1378        BOOL GetResults(PDBPROCESS dbproc)
1379        {
1380           if ( SQLFetch(dbproc->hstmt) == SQL_ERROR )
1381            {
1382                ODBCError(dbproc);
1383               if ( *((BOOL *)dbgetuserdata(dbproc)) )
1384                    return FALSE;
1385               return TRUE;
1386            }
1387            return FALSE;
1388        }
1389
1390        /* FUNCTION: BOOL MoreResults(DBPROCESS
   dbproc)
1391         *
1392         * PURPOSE: This function wraps the odbc SQL-
   MoreResults API so that error handling and
1393         *          and deadlock are taken care of in
   a common location.
1394         *
1395         * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
   cess structure
1396         *
1397         * RETURNS:     none
1398         *
1399         * COMMENTS:    none
1400         *
1401         */
1402
1403        BOOL MoreResults(PDBPROCESS  dbproc)
1404        {
1405            if ( SQLMoreResults(dbproc->hstmt) ==
   SQL_ERROR )
1406            {
1407                ODBCError(dbproc);
1408               if ( *((BOOL *)dbgetuserdata(dbproc)) )
1409                    return FALSE;
1410               return TRUE;
1411            }
1412            return FALSE;
1413        }
1414
1415    #endif
```

delisrv.h

```
 1  /*  FILE:       DELISRV.H, MSTPCC.300
 2   *  -----       ------------------------------------
------
 3   *              Microsoft TPC-C Kit Ver. 3.00.000
 4   *              Audited 08/23/96, By Francois Raab
 5   *
 6   *              Copyright Microsoft, 1996
 7   *
 8   *  PURPOSE:    Header file for delivery service exe-
cutable
 9   *  Author:     Philip Durr
10   *              philipdu@Microsoft.com
11   */
12
13  #define AVAILABLE      0                    //queue
 array element available
14  #define WRITE_LOCKED    1                   //queue
 array element is being written to
15  #define READ_LOCKED     2                   //queue
 array element is begin read
16  #define INUSE          4                    //queue
 array element has information stored in it
17
18  #define CTRL_C         3                   //<Ctrl>
 C, exit key code
19
20  #define DEFCLPACKSIZE                  4096
    //default DB Library SQL Connection pack size
21
22  #define ERR_SUCCESS                    0       //Suc-
cess, no error.
23  #define ERR_CANNOT_CREATE_THREAD       1000    //Can-
not create thread.
24  #define ERR_DBGETDATA_FAILED           1001    //Get
data failed.
25  #define ERR_REGISTRY_NOT_SETUP         1002    //Reg-
istry not setup for tpcc.
26  #define ERR_CANNOT_ACCESS_DELIVERY_FN   1003
//Cannot access ReadDelivery cache.
27  #define ERR_CANNOT_ACCESS_REGISTRY      1004
//Cannot access registry key TPCC.
28  #define ERR_CANNOT_CREATE_RESULTS_FILE  1005
//Cannot create results file.
29  #define ERR_CANNOT_OPEN_PIPE           1006    //Can-
not open delivery pipe.
30  #define ERR_READ_PIPE                 1007   //Error
reading pipe
31  #define ERR_INSUFFICIENT_MEMORY         1008
//insufficient memory
32  #define ERR_ODBC_SQLALLOCENV           1009    //Can-
not allocated ODBC env handle
33  #define ERR_SQL_ATTR_ODBC_VERSION       1010
//Cannot set ODBC version
34  #define ERR_SQL_ATTR_CONNECTION_POOLING 1011
//Cannot set Connection Pooling
35
36  typedef struct _DELIVERY_TRANSACTION
37  {
38      SYSTEMTIME  queue;          //time delivery trans-
action queued
39      short       w_id;           //delivery warehouse
40      short       o_carrier_id;   //carrier id
41  } DELIVERY_TRANSACTION;
42
43  typedef DELIVERY_TRANSACTION
*LPDELIVERY_TRANSACTION;   //pointer to delivery trans-
action queue
44
45  typedef struct _DELIVERY_PACKET
46  {
47      BOOL                    bInUse;         //entry
current in use
48      OVERLAPPED              ov;             //pipe io
overlapped structure
49      DELIVERY_TRANSACTION    trans;         //delivery
transaction information
50  } DELIVERY_PACKET, *LPDELIVERY_PACKET;
51
52  typedef struct _SERRORMSG
53  {
54      int     iError;         //error mesage id
55      char    szMsg[80];      //error message
56  } SERRORMSG;
57
58  #ifdef USE_ODBC
59      typedef struct _DBPROCESS
60      {
61          HDBC    hdbc;
62          HSTMT   hstmt;
63          int     spid;
64          void    *uPtr;
65      } DBPROCESS, *PDBPROCESS;
66
67      //dblib error message return values
68      #define INT_EXIT        0
69      #define INT_CONTINUE    1
70      #define INT_CANCEL      2
71  #endif
72
73  //delivery transaction structure
74  typedef struct DELIVERY
75  {
76      short       w_id;           //warehouse id
77      short       o_carrier_id;   //carrier id
```

```
 78      int         spid;           //db library spid
 79      long        o_id[10];       //returned delivery
 transaction ids
 80      DBPROCESS   *dbproc;        //db library DBPROCESS
 pointer
 81      SYSTEMTIME  queue;          //delivery transaction
 queue time
 82      SYSTEMTIME  trans_end;      //delivery transaction
 finished time
 83   } DELIVERY;
 84
 85   typedef DELIVERY *LPDELIVERY;   //pointer to delivery
 structure
 86
 87   //function prototypes
 88   void            main(int argc, char *argv[]);
 89   static  void    cls(void);
 90   static  int     RunDelivery(void);
 91   static  void    QuitStatus(void);
 92   static  void    AnimateWait1(void);
 93   static  void    AnimateWait(void);
 94   static  int     Init(void);
 95   static  void    Restore(void);
 96   static  void    ErrorMessage(int iError);
 97   static  BOOL    GetParameters(int argc, char
  *argv[]);
 98   static  void    PrintParameters(void);
 99   static  void    PrintHeader(void);
100   static  int     ReadRegistrySettings(void);
101   static  void    CheckKey(void *ptr);
102   static  void    DeliveryHandler( void *ptr );
103   static  void    DeliveryThread( void *ptr );
104
105   #ifndef USE_ODBC
106      static  int    err_handler(DBPROCESS *dbproc, int
  severity, int dberr, int oserr, char *dberrstr, char
  *oserrstr);
107   #endif
108
109   #ifdef USE_ODBC
110      #define DBINT   int
111   #endif
112
113   static  int     msg_handler(DBPROCESS *dbproc, DBINT
  msgno, int msgstate, int severity, char *msgtext);
114   static  BOOL    SQLOpenConnection(DBPROCESS **dbproc,
  char *server, char *database, char *user, char *pass-
  word, int *spid);
115   static  void    WriteLog(LPDELIVERY pDelivery);
116   static  void    CalculateElapsedTime(int *pElapsed,
  LPSYSTEMTIME lpBegin, LPSYSTEMTIME lpEnd);
117   static  int     SQLDelivery(DELIVERY *pDelivery);
118   static  BOOL    SQLDetectDeadlock(DBPROCESS *dbproc);
119   static  BOOL    ReadDeliveryInfo(short *w_id, short
  *o_carrier_id);
120   static  BOOL    PostDeliveryInfo(short w_id, short
  o_carrier_id);
121   static  int     OpenLogFile(void);
122
123   #ifdef USE_ODBC
124      void dbsetuserdata(PDBPROCESS dbproc, void
  *uPtr);
125      void *dbgetuserdata(PDBPROCESS dbproc);
126      void BindParameter(PDBPROCESS dbproc, UWORD ipar,
  SWORD fCType, SWORD fSqlType, UDWORD cbColDef, SWORD
  ibScale, PTR rgbValue, SDWORD cbValueMax);
127      void ODBCError(PDBPROCESS dbproc);
128      BOOL ExecuteStatement(PDBPROCESS dbproc, char
  *szStatement);
129      BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT
  icol, SQLSMALLINT fCType, SQLPOINTER rgbValue, SQLINTE-
  GER cbValueMax, SDWORD *piLength);
130      BOOL GetResults(PDBPROCESS dbproc);
131      BOOL MoreResults(PDBPROCESS  dbproc);
132      BOOL ReopenConnection(PDBPROCESS dbproc);
133   #endif
```

dll.mak

```
1      !IF "$(CFG)" == ""
 2   CFG=Debug
 3   !MESSAGE No configuration specified.  Defaulting to
 Debug
 4   !ENDIF
 5
 6   OUT_PATH    = c:\temp\mckee\objs\tpcc\dll
 7
 8   ALL:    $(OUT_PATH)\. dlls
 9
10   $(OUT_PATH)\.:
11       if not exist $(OUT_PATH) md $(OUT_PATH)
12
13   dlls:original_dll tuxedo_process_dll
 tuxedo_threads_dll
14
15   original_dll:
16       cd original
17       nmake CFG=$(CFG) /$(MAKEFLAGS) /f original.mak
18       cd ..
19
20   tuxedo_process_dll:
21       cd tuxedo_process
22       nmake CFG=$(CFG) /$(MAKEFLAGS) /f
 tuxedo_process.mak
23       cd ..
24
25   tuxedo_threads_dll:
26       cd tuxedo_threads
27       nmake CFG=$(CFG) /$(MAKEFLAGS) /f
 tuxedo_threads.mak
28       cd ..
```

error.c

```
 1   #include <windows.h>
 2   #include <string.h>
 3   #include <stdio.h>
 4   #include "trans.h"
 5   #include "tpcc.h"
 6   #include "util.h"
 7   #include "error.h"
 8
 9
10   /* FUNCTION: void ErrorMes-
 sage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int iEr-
 rorType, char *szMsg)
11    *
12    * PURPOSE: This function displays an error message
 in the client browser.
13    *
14    * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 passed in structure pointer from inetsrv.
15    *              int             iError          id of
 error message
16    *              int             iErrorType      error
 type, ERR_TYPE_SQL, ERR_TYPE_DBLIB, or ERR_TYPE_WEBDLL
17    *            int            iTermId        terminal
 id from browser
18    *              int             iSyncid         sync
 id from browser
19    *            char *         szMsg          optional
 error message string used with ERR_TYPE_SQL and
20    *
 ERR_TYPE_DBLIB
21    *
22    * RETURNS:     None
23    *
24    * COMMENTS:    If the error type is ERR_TYPE_WEBDLL
 the szmsg parameter may be NULL because it
25    *              is ignored. If the error type is
 ERR_TYPE_SQL or ERR_TYPE_DBLIB then the szMsg
26    *              parameter contains the text of the
 error message, so the szMsg parameter cannot
27    *              be NULL.
```

```
28   *
29   */
30
31  void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int
iError, int iErrorType, char *szMsg, int iTermId, int
iSyncId)
32  {
33      int i;
34
35      static SERRORMSG errorMsgs[] =
36      {
37          {   ERR_SUCCESS,
"Success, no
error."                                                 },
38          {   ERR_COMMAND_UNDEFINED,
"Command unde-
fined."                                                 },
39          {   ERR_NOT_IMPLEMENTED_YET,
"Not Implemented
Yet."                                                 },
40          {   ERR_CANNOT_INIT_TERMINAL,
"Cannot initialize client connec-
tion."                                },
41          {   ERR_OUT_OF_MEMORY,
"insufficient mem-
ory."                                                 },
42          {   ERR_NEW_ORDER_NOT_PROCESSED,
"Cannot process new Order
form."                                          },
43          {   ERR_PAYMENT_NOT_PROCESSED,
"Cannot process payment
form."                                           },
44          {   ERR_NO_SERVER_SPECIFIED,              "No
Server name speci-
fied."                                                },
45          {   ERR_ORDER_STATUS_NOT_PROCESSED,
"Cannot process order status
form."                                          },
46          {   ERR_W_ID_INVALID,
"Invalid Warehouse
ID."                                                  },
47          {   ERR_CAN_NOT_SET_MAX_CONNECTIONS,
"Insufficient memory to allocate # connec-
tions."                     },
48          {   ERR_NOSUCH_CUSTOMER,                  "No
such cus-
tomer."                                                 },
49          {   ERR_D_ID_INVALID,
"Invalid District ID Must be 1 to
10."                                  },
50          {   ERR_MAX_CONNECT_PARAM,
"Max client connections exceeded, run install to
increase."          },
51          {   ERR_INVALID_SYNC_CONNECTION,
"Invalid Terminal Sync
ID."                                               },
52          {   ERR_INVALID_TERMID,
"Invalid Terminal
ID."                                                  },
53          {   ERR_PAYMENT_INVALID_CUSTOMER,
"Payment Form, No such Cus-
tomer."                                         },
54          {   ERR_SQL_OPEN_CONNECTION,
"SQLOpenConnection API
Failed."                                          },
55          {   ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,
"Stock Level missing Threshold key
\"TT*\"."                                 },
56          {   ERR_STOCKLEVEL_THRESHOLD_INVALID,
"Stock Level Threshold invalid data type range = 1 - 99."
},
57          {   ERR_STOCKLEVEL_THRESHOLD_RANGE,
"Stock Level Threshold out of range, range must be 1 -
99."          },
58          {   ERR_STOCKLEVEL_NOT_PROCESSED,
"Stock Level not pro-
cessed."                                             },
59          {   ERR_NEWORDER_FORM_MISSING_DID,
"New Order missing District key
\"DID*\"."                                  },
60          {   ERR_NEWORDER_DISTRICT_INVALID,
"New Order District ID Invalid range 1 -
10."                        },
61          {   ERR_NEWORDER_DISTRICT_RANGE,
"New Order District ID out of Range. Range = 1 -
10."                },
62          {   ERR_NEWORDER_CUSTOMER_KEY,
"New Order missing Customer key
\"CID*\"."                                  },
63          {   ERR_NEWORDER_CUSTOMER_INVALID,
"New Order customer id invalid data type, range = 1 to
3000."        },
64          {   ERR_NEWORDER_CUSTOMER_RANGE,
"New Order customer id out of range, range = 1 to 3000."
},
65          {   ERR_NEWORDER_MISSING_IID_KEY,
"New Order missing Item Id key
\"IID*\"."                                   },
66          {   ERR_NEWORDER_ITEM_BLANK_LINES,
"New Order blank order lines all orders must be continu-
ous."         },
67          {   ERR_NEWORDER_ITEMID_INVALID,
"New Order Item Id is wrong data type, must be numeric."
},
68          {   ERR_NEWORDER_MISSING_SUPPW_KEY,
"New Order missing Supp_W key
\"SP##*\"."                                },
69          {   ERR_NEWORDER_SUPPW_INVALID,
"New Order Supp_W invalid data type must be
numeric."                },
70          {   ERR_NEWORDER_MISSING_QTY_KEY,
"New Order Missing Qty key
\"Qty##*\"."                                 },
71          {   ERR_NEWORDER_QTY_INVALID,
"New Order Qty invalid must be numeric range 1 -
99."              },
72          {   ERR_NEWORDER_SUPPW_RANGE,
"New Order Supp_W value out of range range = 1 - Max
Warehouses."    },
73          {   ERR_NEWORDER_ITEMID_RANGE,
"New Order Item Id is out of range. Range = 1 to 999999."
},
74          {   ERR_NEWORDER_QTY_RANGE,
"New Order Qty is out of range. Range = 1 to
99."                    },
75          {   ERR_PAYMENT_DISTRICT_INVALID,
"Payment District ID is invalid must be 1 -
10."                    },
76          {   ERR_NEWORDER_SUPPW_WITHOUT_ITEMID,
"New Order Supp_W field entered without a corrisponding
Item_Id."    },
77          {   ERR_NEWORDER_QTY_WITHOUT_ITEMID,
"New Order Qty entered without a corrisponding Item_Id."
},
78          {   ERR_NEWORDER_NOITEMS_ENTERED,
"New Order Blank Items between items, items must be con-
tinuous."     },
79          {   ERR_PAYMENT_MISSING_DID_KEY,
"Payment missing District Key
\"DID*\"."                                  },
80          {   ERR_PAYMENT_DISTRICT_RANGE,
"Payment District Out of range, range = 1 -
10."                    },
81          {   ERR_PAYMENT_MISSING_CID_KEY,
"Payment missing Customer Key
\"CID*\"."                                   },
82          {   ERR_PAYMENT_CUSTOMER_INVALID,
"Payment Customer data type invalid, must be
numeric."              },
83          {   ERR_PAYMENT_MISSING_CLT,
"Payment missing Customer Last Name Key
\"CLT*\"."                    },
84          {   ERR_PAYMENT_LAST_NAME_TO_LONG,
"Payment Customer last name longer than 16 charac-
```

```
ters."             },
85          {   ERR_PAYMENT_CUSTOMER_RANGE,
"Payment Customer ID out of range, must be 1 to
3000."             },
86          {   ERR_PAYMENT_CID_AND_CLT,
"Payment Customer ID and Last Name entered must be one or
other."   },
87          {   ERR_PAYMENT_MISSING_CDI_KEY,
"Payment missing Customer district key
\"CDI*\"."                  },
88          {   ERR_PAYMENT_CDI_INVALID,
"Payment Customer district invalid must be
numeric."               },
89          {   ERR_PAYMENT_CDI_RANGE,
"Payment Customer district out of range must be 1 - 10."
},
90          {   ERR_PAYMENT_MISSING_CWI_KEY,
"Payment missing Customer Warehouse key
\"CWI*\"."                  },
91          {   ERR_PAYMENT_CWI_INVALID,
"Payment Customer Warehouse invalid must be
numeric."               },
92          {   ERR_PAYMENT_CWI_RANGE,
"Payment Customer Warehouse out of range, 1 to Max Ware-
houses."     },
93          {   ERR_PAYMENT_MISSING_HAM_KEY,
"Payment missing Amount key
\"HAM*\"."                              },
94          {   ERR_PAYMENT_HAM_INVALID,
"Payment Amount invalid data type must be
numeric."                },
95          {   ERR_PAYMENT_HAM_RANGE,
"Payment Amount out of range, 0 -
9999.99."                           },
96          {   ERR_ORDERSTATUS_MISSING_DID_KEY,
"Order Status missing District key
\"DID*\"."                          },
97          {   ERR_ORDERSTATUS_DID_INVALID,
"Order Status District invalid, value must be numeric 1 -
10."      },
98          {   ERR_ORDERSTATUS_DID_RANGE,
"Order Status District out of range must be 1 -
10."               },
99          {   ERR_ORDERSTATUS_MISSING_CID_KEY,
"Order Status missing Customer key
\"CID*\"."                          },
100         {   ERR_ORDERSTATUS_MISSING_CLT_KEY,
"Order Status missing Customer Last Name key
\"CLT*\"."              },
101         {   ERR_ORDERSTATUS_CLT_RANGE,
"Order Status Customer last name longer than 16 charac-
ters."        },
102         {   ERR_ORDERSTATUS_CID_INVALID,
"Order Status Customer ID invalid, range must be numeric
1 - 3000." },
103         {   ERR_ORDERSTATUS_CID_RANGE,
"Order Status Customer ID out of range must be 1 - 3000."
},
104         {   ERR_ORDERSTATUS_CID_AND_CLT,
"Order Status Customer ID and LastName entered must be
only one."   },
105         {   ERR_DELIVERY_MISSING_OCD_KEY,
"Delivery missing Carrier ID key
\"OCD*\"."                            },
106         {   ERR_DELIVERY_CARRIER_INVALID,
"Delivery Carrier ID invalid must be numeric 1 -
10."               },
107         {   ERR_DELIVERY_CARRIER_ID_RANGE,
"Delivery Carrier ID out of range must be 1 -
10."                  },
108         {   ERR_PAYMENT_MISSING_CLT_KEY,
"Payment missing Customer Last Name key
\"CLT*\"."                  },
109     { 0,                ""
}
110     };
111
```

```
112     static char szNoMsg[] = "";
113     char        *szForm;
114
115     if ( !szMsg  )
116         szMsg = szNoMsg;
117
118     if ( iTermId > 0 && IsValidTermId(iTermId) )
119         szForm = Term.pClientData[iTermId].szBuffer;
 //if termid valid use common terminal static buffer.
120     else
121         szForm = Term.pClientData[0].szBuffer;
 //else term id invalid so use common terminal static
 buffer.
122
123
124     switch(iErrorType)
125     {
126         case ERR_TYPE_WEBDLL:
127             for(i=0; errorMsgs[i].szMsg[0]; i++)
128             {
129                 if ( iError == errorMsgs[i].iError )
130                     break;
131             }
132             if ( !errorMsgs[i].szMsg[0] )
133                 i = 1;
134             strcpy(szForm,  "<HTML><HEAD><TITLE>Wel-
 come To TPC-C</TITLE></HEAD><BODY><FORM
 ACTION=\"tpcc.dll\" METHOD=\"GET\">");
135             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iEr-
 rorType);
136             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
 mId);
137             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
 cId);
138             wsprintf(szForm+strlen(szForm), "Error:
 TPCCWEB(%d): %s", iError, errorMsgs[i].szMsg);
139             strcat(szForm, "</FORM><BODY></HTML>");
140             WriteZString(pECB, szForm);
141             break;
142         case ERR_TYPE_SQL:
143             strcpy(szForm,  "<HTML><HEAD><TITLE>Wel-
 come To TPC-C</TITLE></HEAD><BODY><FORM
 ACTION=\"tpcc.dll\" METHOD=\"GET\">");
144             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iEr-
 rorType);
145             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
 mId);
146             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
 cId);
147             wsprintf(szForm+strlen(szForm), "Error:
 SQLSVR(%d): %s", iError, szMsg);
148             strcat(szForm, "</FORM><BODY></HTML>");
149             WriteZString(pECB, szForm);
150             break;
151         case ERR_TYPE_DBLIB:
152             strcpy(szForm,  "<HTML><HEAD><TITLE>Wel-
 come To TPC-C</TITLE></HEAD><BODY><FORM
 ACTION=\"tpcc.dll\" METHOD=\"GET\">");
153             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iEr-
 rorType);
154             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
 mId);
155             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
 cId);
156             wsprintf(szForm+strlen(szForm), "Error:
 DBLIB(%d): %s", iError, szMsg);
157             strcat(szForm, "</FORM><BODY></HTML>");
158             WriteZString(pECB, szForm);
```

```
159             break;
160          case ERR_TYPE_ODBC:
161             strcpy(szForm,  "<HTML><HEAD><TITLE>Wel-
 come To TPC-C</TITLE></HEAD><BODY><FORM
 ACTION=\"tpcc.dll\" METHOD=\"GET\">");
162             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iEr-
 rorType);
163             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
 mId);
164             wsprintf(szForm+strlen(szForm), "<INPUT
 TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
 cId);
165             wsprintf(szForm+strlen(szForm), "Error:
 ODBC(%d): %s", iError, szMsg);
166             strcat(szForm, "</FORM><BODY></HTML>");
167             WriteZString(pECB, szForm);
168             break;
169      }
170      return;
171  }
```

error.h

```
 1    #ifndef ERROR_H_INCLUDED
 2    #define ERROR_H_INCLUDED
 3
 4    extern TERM Term;
 5
 6    //error message structure used in ErrorMessage API
 7    typedef struct _SERRORMSG
 8    {
 9        int     iError;            //error id of message
10        char    szMsg[80];         //message to sent to
browser
11    } SERRORMSG;
12
13    void WriteZString(EXTENSION_CONTROL_BLOCK *pECB,
char *szStr);
14    void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB,
int iError, int iErrorType, char *szMsg, int iTermId, int
iSyncId);
15
16    #define ERR_BAD_ITEM_ID                        1
//expected abort record in txnRecord
17    #define ERR_TYPE_DELIVERY_POST
2        //expected delivery post failed
18    #define ERR_TYPE_WEBDLL                        3
//tpcc web generated error
19    #define ERR_TYPE_SQL                           4
//sql server generated error
20    #define ERR_TYPE_DBLIB                         5
//dblib generated error
21    #define ERR_TYPE_ODBC                          6
//odbc generated error
22    #define ERR_TYPE_SOCKET                        7
//error on communication socket client rte only
23    #define ERR_TYPE_DEADLOCK
8        //dblib and odbc only deadlock condition
24
25    #define ERR_SUCCESS                           1000
//"Success, no error.
26    #define ERR_COMMAND_UNDEFINED                 1001
//"Command undefined.
27    #define ERR_NOT_IMPLEMENTED_YET               1002
//"Not Implemented Yet.
28    #define ERR_CANNOT_INIT_TERMINAL              1003
//"Cannot initialize client connection.
29    #define ERR_OUT_OF_MEMORY                     1004
//"insufficient memory.
30    #define ERR_NEW_ORDER_NOT_PROCESSED           1005
//"Cannot process new Order form.
31    #define ERR_PAYMENT_NOT_PROCESSED             1006
//"Cannot process payment form.
32    #define ERR_NO_SERVER_SPECIFIED               1007
//"No Server name specified.
33    #define ERR_ORDER_STATUS_NOT_PROCESSED        1008
//"Cannot process order status form.
34    #define ERR_W_ID_INVALID                      1009
//"Invalid Warehouse ID.
35    #define ERR_CAN_NOT_SET_MAX_CONNECTIONS       1010
//"Insufficient memory to allocate # connections.
36    #define ERR_NOSUCH_CUSTOMER                   1011
//"No such customer.
37    #define ERR_D_ID_INVALID                      1012
//"Invalid District ID Must be 1 to 10.
38    #define ERR_MAX_CONNECT_PARAM                 1013
//"Max client connections exceeded, run install to
increase.
39    #define ERR_INVALID_SYNC_CONNECTION           1014
//"Invalid Terminal Sync ID.
40    #define ERR_INVALID_TERMID                    1015
//"Invalid Terminal ID.
41    #define ERR_PAYMENT_INVALID_CUSTOMER          1016
//"Payment Form, No such Customer.
42    #define ERR_SQL_OPEN_CONNECTION               1017
//"SQLOpenConnection API Failed.
43    #define ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY
1018     //"Stock Level missing Threshold key "TT*".
44    #define ERR_STOCKLEVEL_THRESHOLD_INVALID      1019
//"Stock Level Threshold invalid data type range = 1 -
99.
45    #define ERR_STOCKLEVEL_THRESHOLD_RANGE        1020
//"Stock Level Threshold out of range, range must be 1 -
99.
46    #define ERR_STOCKLEVEL_NOT_PROCESSED          1021
//"Stock Level not processed.
47    #define ERR_NEWORDER_FORM_MISSING_DID         1022
//"New Order missing District key "DID*".
48    #define ERR_NEWORDER_DISTRICT_INVALID         1023
//"New Order District ID Invalid range 1 - 10.
49    #define ERR_NEWORDER_DISTRICT_RANGE           1024
//"New Order District ID out of Range. Range = 1 - 10.
50    #define ERR_NEWORDER_CUSTOMER_KEY             1025
//"New Order missing Customer key "CID*".
51    #define ERR_NEWORDER_CUSTOMER_INVALID         1026
//"New Order customer id invalid data type, range = 1 to
3000.
52    #define ERR_NEWORDER_CUSTOMER_RANGE           1027
//"New Order customer id out of range, range = 1 to 3000.
53    #define ERR_NEWORDER_MISSING_IID_KEY          1028
//"New Order missing Item Id key "IID*".
54    #define ERR_NEWORDER_ITEM_BLANK_LINES         1029
//"New Order blank order lines all orders must be contin-
uous.
55    #define ERR_NEWORDER_ITEMID_INVALID           1030
//"New Order Item Id is wrong data type, must be numeric.
56    #define ERR_NEWORDER_MISSING_SUPPW_KEY        1031
//"New Order missing Supp_W key "SP##*".
57    #define ERR_NEWORDER_SUPPW_INVALID            1032
//"New Order Supp_W invalid data type must be numeric.
58    #define ERR_NEWORDER_MISSING_QTY_KEY          1033
//"New Order Missing Qty key "Qty##*".
59    #define ERR_NEWORDER_QTY_INVALID              1034
//"New Order Qty invalid must be numeric range 1 - 99.
60    #define ERR_NEWORDER_SUPPW_RANGE              1035
//"New Order Supp_W value out of range range = 1 - Max
Warehouses.
61    #define ERR_NEWORDER_ITEMID_RANGE             1036
//"New Order Item Id is out of range. Range = 1 to
999999.
62    #define ERR_NEWORDER_QTY_RANGE                1037
//"New Order Qty is out of range. Range = 1 to 99.
63    #define ERR_PAYMENT_DISTRICT_INVALID          1038
//"Payment District ID is invalid must be 1 - 10.
64    #define ERR_NEWORDER_SUPPW_WITHOUT_ITEMID     1039
//"New Order Supp_W field entered without a corrisponding
Item_Id.
65    #define ERR_NEWORDER_QTY_WITHOUT_ITEMID       1040
//"New Order Qty entered without a corrisponding Item_Id.
66    #define ERR_NEWORDER_NOITEMS_ENTERED          1041
//"New Order Blank Items between items, items must be
continuous.
```

```
67   #define ERR_PAYMENT_MISSING_DID_KEY          1042
//"Payment missing District Key "DID*".
68   #define ERR_PAYMENT_DISTRICT_RANGE           1043
//"Payment District Out of range, range = 1 - 10.
69   #define ERR_PAYMENT_MISSING_CID_KEY          1044
//"Payment missing Customer Key "CID*".
70   #define ERR_PAYMENT_CUSTOMER_INVALID         1045
//"Payment Customer data type invalid, must be numeric.
71   #define ERR_PAYMENT_MISSING_CLT              1046
//"Payment missing Customer Last Name Key "CLT*".
72   #define ERR_PAYMENT_LAST_NAME_TO_LONG        1047
//"Payment Customer last name longer than 16 characters.
73   #define ERR_PAYMENT_CUSTOMER_RANGE           1048
//"Payment Customer ID out of range, must be 1 to 3000.
74   #define ERR_PAYMENT_CID_AND_CLT              1049
//"Payment Customer ID and Last Name entered must be one
or other.
75   #define ERR_PAYMENT_MISSING_CDI_KEY          1050
//"Payment missing Customer district key "CDI*".
76   #define ERR_PAYMENT_CDI_INVALID              1051
//"Payment Customer district invalid must be numeric.
77   #define ERR_PAYMENT_CDI_RANGE                1052
//"Payment Customer district out of range must be 1 - 10.
78   #define ERR_PAYMENT_MISSING_CWI_KEY          1053
//"Payment missing Customer Warehouse key "CWI*".
79   #define ERR_PAYMENT_CWI_INVALID              1054
//"Payment Customer Warehouse invalid must be numeric.
80   #define ERR_PAYMENT_CWI_RANGE                1055
//"Payment Customer Warehouse out of range, 1 to Max
Warehouses.
81   #define ERR_PAYMENT_MISSING_HAM_KEY          1056
//"Payment missing Amount key "HAM*".
82   #define ERR_PAYMENT_HAM_INVALID              1057
//"Payment Amount invalid data type must be numeric.
83   #define ERR_PAYMENT_HAM_RANGE                1058
//"Payment Amount out of range, 0 - 9999.99.
84   #define ERR_ORDERSTATUS_MISSING_DID_KEY      1059
//"Order Status missing District key "DID*".
85   #define ERR_ORDERSTATUS_DID_INVALID          1060
//"Order Status District invalid, value must be numeric 1
- 10.
86   #define ERR_ORDERSTATUS_DID_RANGE            1061
//"Order Status District out of range must be 1 - 10.
87   #define ERR_ORDERSTATUS_MISSING_CID_KEY      1062
//"Order Status missing Customer key "CID*".
88   #define ERR_ORDERSTATUS_MISSING_CLT_KEY      1063
//"Order Status missing Customer Last Name key "CLT*".
89   #define ERR_ORDERSTATUS_CLT_RANGE            1064
//"Order Status Customer last name longer than 16 charac-
ters.
90   #define ERR_ORDERSTATUS_CID_INVALID          1065
//"Order Status Customer ID invalid, range must be
numeric 1 - 3000.
91   #define ERR_ORDERSTATUS_CID_RANGE            1066
//"Order Status Customer ID out of range must be 1 -
3000.
92   #define ERR_ORDERSTATUS_CID_AND_CLT          1067
//"Order Status Customer ID and LastName entered must be
only one."
93   #define ERR_DELIVERY_MISSING_OCD_KEY         1068
//"Delivery missing Carrier ID key \"OCD*\".
94   #define ERR_DELIVERY_CARRIER_INVALID         1069
//"Delivery Carrier ID invalid must be numeric 1 - 10.
95   #define ERR_DELIVERY_CARRIER_ID_RANGE        1070
//"Delivery Carrier ID out of range must be 1 - 10.
96   #define ERR_PAYMENT_MISSING_CLT_KEY          1071
//"Payment missing Customer Last Name key "CLT*".
97
98   #endif
```

errorstring.c

```
1  #include <windows.h>
2  #include "errorstring.h"
3
4  void
5  ErrorString(char *buf, int buf_size, DWORD dwError)
6  {
7
8      LPVOID lpMsgBuf;
9
10     int nBytes;
11
12     nBytes = FormatMessage(
13         FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM,
14         NULL,
15         GetLastError(),
16        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), //
Default language
17         buf,
18         buf_size,
19         NULL
20     );
21
22     if (!nBytes)
23     {
24         snprintf(buf, buf_size, "Unable to get error
message for error %d", dwError);
25     }
26 }
```

errorstring.h

```
 1void ErrorString(DWORD dwError, char *buf, int
buf_size);
```

getopt.c

```
1  #ifndef __unix
2  /* got this off net.sources. */
3  #include <stdio.h>
4  #include "getopt.h"
5
6  /*
7   * get option letter from argument vector
8   */
9  int opterr = 1,     /* useless, never set or used */
10     optind = 1,     /* index into parent argv vector
*/
11     optopt;        /* character checked for validity
*/
12 char    *optarg;        /* argument associated with
option */
13
14 #define BADCH   (int)'?'
15 #define NEEDARG (int)':'
16 #define EMSG    ""
17
18
19 getopt(int nargc,char * const * nargv,const char
*ostr)
20 {
21     static char *place = EMSG;  /* option letter pro-
cessing */
22     register char   *oli;       /* option letter list
index */
23     char    *strchr();
24
25     if(!*place) {          /* update scanning pointer
*/
26        if(optind >= nargc || *(place = nargv[optind])
!= '-' || !*++place) return(EOF);
27         if (*place == '-') {    /* found "--" */
28             ++optind;
29             return(EOF);
30         }
31     }              /* option letter okay? */
32     if ((optopt = (int)*place++) == (int)':' || !(oli
= strchr(ostr,optopt))) {
33         if(!*place) ++optind;
34         return (BADCH);
35     }
```

```
36      if (*++oli != ':') {       /* don't need argument
 */
37          optarg = NULL;
38          if (!*place) ++optind;
39      }
40      else {              /* need an argument */
41          if (*place) optarg = place; /* no white space
 */
42          else if (nargc <= ++optind) {   /* no arg */
43              place = EMSG;
44              return(NEEDARG);
45          }
46         else optarg = nargv[optind];    /* white space
 */
47          place = EMSG;
48          ++optind;
49      }
50      return(optopt);        /* dump back option letter
 */
51  }
52
53  #endif
```

getopt.h

```
1   #ifndef _GETOPT_H_INCLUDED
 2  #define _GETOPT_H_INCLUDED
 3
 4  #ifdef __cplusplus
 5  extern "C" {
 6  #endif
 7
 8  extern int optind, optopt;
 9  extern char *optarg;
10  extern int getopt(int argc, char * const *argv, const
char *options);
11
12  #ifdef __cplusplus
13  }  // end of extern "C"
14  #endif
15  #endif
```

httpext.h

```
1   /********
 2  *
 3  *  Copyright (c) 1995  Process Software Corporation
 4  *
 5  *  Copyright (c) 1995  Microsoft Corporation
 6  *
 7  *
 8  *  Module Name  : HttpExt.h
 9  *
10  *  Abstract :
11  *
12  *     This module contains  the structure definitions
and prototypes for the
13  *     version 1.0 HTTP Server Extension interface.
14  *
15  ******************/
16
17  #ifndef _HTTPEXT_H_
18  #define _HTTPEXT_H_
19
20  #include <windows.h>
21
22  #ifdef __cplusplus
23  extern "C" {
24  #endif
25
26  #define   HSE_VERSION_MAJOR           1      // major
version of this spec
27  #define   HSE_VERSION_MINOR           0      // minor
version of this spec
28  #define   HSE_LOG_BUFFER_LEN         80
29  #define   HSE_MAX_EXT_DLL_NAME_LEN  256
30
31  typedef   LPVOID  HCONN;
32
33  // the following are the status codes returned by the
Extension DLL
34
35  #define   HSE_STATUS_SUCCESS                      1
36  #define   HSE_STATUS_SUCCESS_AND_KEEP_CONN        2
37  #define   HSE_STATUS_PENDING                      3
38  #define   HSE_STATUS_ERROR                        4
39
40  // The following are the values to request services
with the ServerSupportFunction.
41  //  Values from 0 to 1000 are reserved for future ver-
sions of the interface
42
43  #define   HSE_REQ_BASE                            0
44  #define   HSE_REQ_SEND_URL_REDIRECT_RESP          (
HSE_REQ_BASE + 1 )
45  #define   HSE_REQ_SEND_URL                        (
HSE_REQ_BASE + 2 )
46  #define   HSE_REQ_SEND_RESPONSE_HEADER            (
HSE_REQ_BASE + 3 )
47  #define   HSE_REQ_DONE_WITH_SESSION               (
HSE_REQ_BASE + 4 )
48  #define   HSE_REQ_END_RESERVED                  1000
49
50  //
51  //  These are Microsoft specific extensions
52  //
53
54  #define   HSE_REQ_MAP_URL_TO_PATH
(HSE_REQ_END_RESERVED+1)
55  #define   HSE_REQ_GET_SSPI_INFO
(HSE_REQ_END_RESERVED+2)
56
57
58  //
59  // passed to GetExtensionVersion
60  //
61
62  typedef struct   _HSE_VERSION_INFO {
63
64      DWORD  dwExtensionVersion;
65      CHAR   lpszExtension-
Desc[HSE_MAX_EXT_DLL_NAME_LEN];
66
67  } HSE_VERSION_INFO, *LPHSE_VERSION_INFO;
68
69  //
70  // passed to extension procedure on a new request
71  //
72  typedef struct _EXTENSION_CONTROL_BLOCK {
73
74      DWORD     cbSize;                 // size of this
struct.
75      DWORD     dwVersion;              // version info
of this spec
76      HCONN     ConnID;                // Context number
not to be modified!
77      DWORD     dwHttpStatusCode;       // HTTP Status
code
78      CHAR      lpszLogData[HSE_LOG_BUFFER_LEN];// null
terminated log info specific to this Extension DLL
79
80      LPSTR     lpszMethod;            // REQUEST_METHOD
81      LPSTR     lpszQueryString;        // QUERY_STRING
82      LPSTR     lpszPathInfo;           // PATH_INFO
83      LPSTR     lpszPathTranslated;     //
PATH_TRANSLATED
84
85      DWORD     cbTotalBytes;           // Total bytes
indicated from client
86      DWORD     cbAvailable;           // Available num-
ber of bytes
87      LPBYTE    lpbData;                // pointer to
cbAvailable bytes
```

```
 88
 89     LPSTR     lpszContentType;        // Content type
of client data
 90
 91     BOOL (WINAPI * GetServerVariable) ( HCONN
hConn,
 92                                        LPSTR       lpsz-
VariableName,
 93                                           LPVOID
lpvBuffer,
 94                                        LPDWORD     lpd-
wSize );
 95
 96     BOOL (WINAPI * WriteClient)  ( HCONN      ConnID,
 97                                      LPVOID     Buffer,
 98                                    LPDWORD    lpdwBytes,
 99                                   DWORD      dwReserved
);
100
101     BOOL (WINAPI * ReadClient)  ( HCONN      ConnID,
102                                   LPVOID     lpvBuffer,
103                                  LPDWORD    lpdwSize );
104
105     BOOL (WINAPI * ServerSupportFunction)( HCONN
hConn,
106                                          DWORD      dwH-
SERRequest,
107                                             LPVOID
lpvBuffer,
108                                             LPDWORD
lpdwSize,
109                                             LPDWORD
lpdwDataType );
110
111 } EXTENSION_CONTROL_BLOCK,
*LPEXTENSION_CONTROL_BLOCK;
112
113 //
114 //  these are the prototypes that must be exported
from the extension DLL
115 //
116
117 BOOL  WINAPI   GetExtensionVersion(
HSE_VERSION_INFO  *pVer );
118 DWORD WINAPI   HttpExtensionProc(
EXTENSION_CONTROL_BLOCK *pECB );
119
120 // the following type declarations is for the server
side
121
122 typedef BOOL  (WINAPI * PFN_GETEXTENSIONVERSION)(
HSE_VERSION_INFO  *pVer );
123 typedef DWORD (WINAPI * PFN_HTTPEXTENSIONPROC )(
EXTENSION_CONTROL_BLOCK *pECB );
124
125 #ifdef __cplusplus
126 }
127 #endif
128
129 #endif  // end definition _HTTPEXT_H_
130
```

install.c

```
 1  /*  FILE:       INSTALL.C
  2  *              Microsoft TPC-C Kit Ver. 3.00.000
  3  *              Audited 08/23/96, By Francois Raab
  4  *
  5  *              Copyright Microsoft, 1996
  6  *
  7  *  PURPOSE:   Automated installation application for
TPC-C Web Kit
  8  *  Author:     Philip Durr
  9  *              philipdu@Microsoft.com
 10  */
 11
 12 #include <windows.h>
 13 #include <direct.h>
 14 #include  <io.h>
 15 #include <stdlib.h>
 16 #include <stdio.h>
 17 #include <commctrl.h>
 18 #include "install.h"
 19
 20 HICON       hIcon;
 21 HINSTANCE   hInst;
 22
 23 DWORD           versionExeMS;
 24 DWORD           versionExeLS;
 25 DWORD           versionExeMM;
 26 DWORD           versionDllMS;
 27 DWORD           versionDllLS;
 28
 29 static  BOOL    bLog;
 30 static  BOOL    bConnectionPooling;
 31 static  int     iThreads;
 32 static  int     iMaxWareHouse;
 33 static  int     iDelayMs;
 34 static  int     iDeadlockRetry;
 35 static  int     iMaxConnections;
 36 static  int     iPoolThreadLimit;
 37 static  int     iThreadTimeout;
 38 static  int     iListenBackLog;
 39 static  int     iAcceptExOutstanding;
 40 static  int     iDllType;
 41
 42 static  int    iMaxPhysicalMemory;    //max physical
memory in MB
 43 static  int     iConnectDelay;
 44 static  char    szVersion[256];
 45
 46 BOOL    CALLBACK    UpdatedDlgProc(HWND hwnd, UINT
uMsg, WPARAM wParam, LPARAM lParam);
 47 BOOL    CALLBACK    MainDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam);
 48 BOOL    CALLBACK    CopyDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam);
 49 static  void        ProcessOK(HWND hwnd, char *szDll-
Path);
 50 static  void        ReadRegistrySettings(void);
 51 static  void       WriteRegistrySettings(char *szDll-
Path);
 52 static  int         CopyFiles(HWND hDlg, char *szDll-
Path);
 53 static  BOOL        GetInstallPath(char *szDllPath);
 54 static  void        GetVersionInfo(char *szDLLPath,
char *szExePath);
 55 static  BOOL        CheckWWWebService(void);
 56 static  BOOL        StartWWWebService(void);
 57 static  BOOL        StopWWWebService(void);
 58 static  void        UpdateDialog(HWND hDlg);
 59
 60 int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE
hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
 61 {
 62     int iRc;
 63
 64     hInst = hInstance;
 65
 66     InitCommonControls();
 67
 68     hIcon = LoadIcon(hInstance, MAKEINTRE-
SOURCE(IDI_ICON1));
 69
 70     iRc = DialogBox(hInstance, MAKEINTRE-
SOURCE(IDD_DIALOG1), GetDesktopWindow(), MainDlgProc);
 71     if ( iRc )
 72             DialogBoxParam(hInstance, MAKEINTRE-
SOURCE(IDD_DIALOG2), GetDesktopWindow(), UpdatedDlg-
Proc, (LPARAM)iRc);
 73     DestroyIcon(hIcon);
 74
 75     return 0;
 76 }
```

```
 77
 78 BOOL CALLBACK UpdatedDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
 79 {
 80     switch(uMsg)
 81     {
 82         case WM_INITDIALOG:
 83             switch(lParam)
 84             {
 85                 case 1:
 86                     SetDlgItemText(hwnd, IDC_RESULTS,
"DBLIB TPC-C WEB Client Installed");
 87                      break;
 88                 case 2:
 89                     SetDlgItemText(hwnd, IDC_RESULTS,
"ODBC TPC-C WEB Client Installed");
 90                      break;
 91                 case 3:
 92                     SetDlgItemText(hwnd, IDC_RESULTS,
"ODBC Connection Pooling TPC-C WEB Client Installed");
 93                      break;
 94             }
 95             return TRUE;
 96         case WM_COMMAND:
 97             if ( wParam == IDOK )
 98                 EndDialog(hwnd, TRUE);
 99             break;
100         default:
101             break;
102     }
103     return FALSE;
104 }
105
106 BOOL CALLBACK MainDlgProc(HWND hwnd, UINT uMsg,
 WPARAM wParam, LPARAM lParam)
107 {
108     PAINTSTRUCT     ps;
109     MEMORYSTATUS    memoryStatus;
110     char            szTmp[256];
111     static char     szDllPath[256];
112     static char     szExePath[256];
113
114     switch(uMsg)
115     {
116         case WM_INITDIALOG:
117             GlobalMemoryStatus(&memoryStatus);
118             iMaxPhysicalMemory  = (memoryStatus.dwTo-
 talPhys/ 1048576);
119
120             if ( GetInstallPath(szDllPath) )
121             {
122                 MessageBox(hwnd, "Error internet ser-
 vice inetsrv is not installed.", NULL, MB_ICONSTOP |
 MB_OK);
123                 EndDialog(hwnd, FALSE);
124                 return TRUE;
125             }
126
127             bLog                    = FALSE;
128             iThreads                = 4;
129             iMaxWareHouse           = 500;
130             iDelayMs                = 500;
131             iDeadlockRetry          = 3;
132             iMaxConnections         = 25;
133             iPoolThreadLimit        = iMaxPhysicalMem-
 ory * 2;
134             iThreadTimeout          = 86400;
135             iListenBackLog          = 15;
136             iAcceptExOutstanding    = 40;
137             iDllType                = IDC_DBLIB;
138             bConnectionPooling      = FALSE;
139
140             ReadRegistrySettings();
141
142             GetModuleFileName(hInst, szExePath,
 sizeof(szExePath));
143             GetVersionInfo(szDllPath, szExePath);
144             if ( bLog )
145                 CheckDlgButton(hwnd, BN_LOG, 1);
146
147             wsprintf(szTmp, "Version %d.%2.2d.%3.3d",
 versionExeMS, versionExeMM, versionExeLS);
148             SetDlgItemText(hwnd, IDC_VERSION, szTmp);
149
150             SetDlgItemText(hwnd, IDC_PATH, szDllPath);
151             SetDlgItemInt(hwnd, ED_MAXWARE, iMaxWare-
 House, FALSE);
152             SetDlgItemInt(hwnd, ED_THREADS, iThreads,
 FALSE);
153             SetDlgItemInt(hwnd, ED_MAXCONNECTION,
 iMaxConnections, FALSE);
154             SetDlgItemInt(hwnd,
 ED_IIS_MAX_THEAD_POOL_LIMIT, iPoolThreadLimit, FALSE);
155             SetDlgItemInt(hwnd, ED_IIS_THREAD_TIMEOUT,
 iThreadTimeout, FALSE);
156             SetDlgItemInt(hwnd, ED_IIS_LISTEN_BACKLOG,
 iListenBackLog, FALSE);
157             SetDlgItemInt(hwnd,
 ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE, iAcceptExOutstand-
 ing, FALSE);
158             SetDlgItemInt(hwnd,
 ED_USER_CONNECT_DELAY_TIME, iConnectDelay, FALSE);
159
160             if ( !strcmp(szVersion, "DBLIB") )
161             {
162                 CheckDlgButton(hwnd, IDC_DBLIB, 1);
163                 CheckDlgButton(hwnd, IDC_ODBC, 0);
164               CheckDlgButton(hwnd, IDC_CONNECT_POOL,
 0);
165                 EnableWindow(GetDlgItem(hwnd,
 IDC_CONNECT_POOL), FALSE);
166                 EnableWindow(GetDlgItem(hwnd,
 ED_USER_CONNECT_DELAY_TIME), FALSE);
167             }
168             else
169             {
170                 CheckDlgButton(hwnd, IDC_DBLIB, 0);
171                 CheckDlgButton(hwnd, IDC_ODBC, 1);
172               CheckDlgButton(hwnd, IDC_CONNECT_POOL,
 bConnectionPooling);
173                 EnableWindow(GetDlgItem(hwnd,
 IDC_CONNECT_POOL), TRUE);
174                 EnableWindow(GetDlgItem(hwnd,
 ED_USER_CONNECT_DELAY_TIME), TRUE);
175             }
176
177             return TRUE;
178         case WM_PAINT:
179             if ( IsIconic(hwnd) )
180             {
181                 BeginPaint(hwnd, &ps);
182                 DrawIcon(ps.hdc, 0, 0, hIcon);
183                 EndPaint(hwnd, &ps);
184                 return TRUE;
185             }
186             break;
187         case WM_COMMAND:
188             if ( HIWORD(wParam) == BN_CLICKED )
189             {
190                 switch( LOWORD(wParam) )
191                 {
192                     case IDC_DBLIB:
193                         bConnectionPooling = IsDlgBut-
 tonChecked(hwnd, IDC_CONNECT_POOL);
194                          CheckDlgButton(hwnd,
 IDC_CONNECT_POOL, 0);
195                         EnableWindow(GetDlgItem(hwnd,
 IDC_CONNECT_POOL), FALSE);
196                         EnableWindow(GetDlgItem(hwnd,
 ED_USER_CONNECT_DELAY_TIME), FALSE);
197                          return TRUE;
198                     case IDC_ODBC:
199                         EnableWindow(GetDlgItem(hwnd,
 IDC_CONNECT_POOL), TRUE);
```

```
200                         CheckDlgButton(hwnd,
 IDC_CONNECT_POOL, bConnectionPooling);
201                         EnableWindow(GetDlgItem(hwnd,
 ED_USER_CONNECT_DELAY_TIME), bConnectionPooling);
202                         return TRUE;
203                     case IDC_CONNECT_POOL:
204                         EnableWindow(GetDlgItem(hwnd,
 ED_USER_CONNECT_DELAY_TIME), IsDlgButtonChecked(hwnd,
 IDC_CONNECT_POOL) );
205                         return TRUE;
206                     case IDOK:
207                         ProcessOK(hwnd, szDllPath);
208                         return TRUE;
209                     case IDCANCEL:
210                         EndDialog(hwnd, FALSE);
211                         return TRUE;
212                     default:
213                         return FALSE;
214                 }
215             }
216             break;
217         default:
218             break;
219     }
220     return FALSE;
221 }
222
223 static void ProcessOK(HWND hwnd, char *szDllPath)
224 {
225     int     d;
226     HWND    hDlg;
227     int     rc;
228
229     if ( IsDlgButtonChecked(hwnd, BN_LOG) )
230         bLog = TRUE;
231     else
232         bLog = FALSE;
233     iThreads  = GetDlgItemInt(hwnd, ED_THREADS, &d,
 FALSE);
234     iMaxWareHouse = GetDlgItemInt(hwnd, ED_MAXWARE,
 &d, FALSE);
235     iMaxConnections = GetDlgItemInt(hwnd,
 ED_MAXCONNECTION, &d, FALSE);
236
237     iPoolThreadLimit = GetDlgItemInt(hwnd,
 ED_IIS_MAX_THEAD_POOL_LIMIT, &d, FALSE);
238     iThreadTimeout = GetDlgItemInt(hwnd,
 ED_IIS_THREAD_TIMEOUT, &d, FALSE);
239     iListenBackLog = GetDlgItemInt(hwnd,
 ED_IIS_LISTEN_BACKLOG, &d, FALSE);
240     iAcceptExOutstanding = GetDlgItemInt(hwnd,
 ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE, &d, FALSE);
241
242     if ( IsDlgButtonChecked(hwnd, IDC_DBLIB) )
243         iDllType = IDC_DBLIB;
244
245     if ( IsDlgButtonChecked(hwnd, IDC_ODBC) )
246         iDllType = IDC_ODBC;
247
248     if ( IsDlgButtonChecked(hwnd, IDC_CONNECT_POOL) )
249         bConnectionPooling = TRUE;
250     else
251         bConnectionPooling = FALSE;
252
253     iConnectDelay = GetDlgItemInt(hwnd,
 ED_USER_CONNECT_DELAY_TIME, &d, FALSE);
254
255     ShowWindow(hwnd, SW_HIDE);
256     hDlg = CreateDialog(hInst, MAKEINTRE-
 SOURCE(IDD_DIALOG3), hwnd, CopyDlgProc);
257     ShowWindow(hDlg, SW_SHOWNA);
258     UpdateDialog(hDlg);
259     rc = CopyFiles(hDlg, szDllPath);
260     if ( !rc )
261     {
262         ShowWindow(hwnd, SW_SHOWNA);
263         DestroyWindow(hDlg);
264         MessageBox(hwnd, "Error(s) occured when creat-
 ing tpcc.dll", NULL, MB_ICONSTOP | MB_OK);
265         EndDialog(hwnd, 0);
266         return;
267     }
268     SetDlgItemText(hDlg, IDC_STATUS, "Updating Regis-
 try.");
269     SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
270     UpdateDialog(hDlg);
271
272     if ( iDllType == IDC_DBLIB )
273     {
274         strcpy(szVersion, "DBLIB");
275         rc = 1;
276     }
277     else if (!bConnectionPooling )
278     {
279         strcpy(szVersion, "ODBC");
280         rc = 2;
281     }
282     else
283     {
284         strcpy(szVersion, "ODBC");
285         rc = 3;
286     }
287
288     WriteRegistrySettings(szDllPath);
289
290     Sleep(100);
291
292     ShowWindow(hwnd, SW_SHOWNA);
293     DestroyWindow(hDlg);
294
295     EndDialog(hwnd, rc);
296     return;
297 }
298
299 static void ReadRegistrySettings(void)
300 {
301     HKEY    hKey;
302     DWORD   size;
303     DWORD   type;
304     char    szTmp[256];
305
306     if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFT-
 WARE\\Microsoft\\TPCC", 0, KEY_READ, &hKey) ==
 ERROR_SUCCESS )
307     {
308         size = sizeof(szTmp);
309
310         bLog = FALSE;
311         if ( RegQueryValueEx(hKey, "LOG", 0, &type,
 szTmp, &size) == ERROR_SUCCESS )
312             if ( !stricmp(szTmp, "ON") )
313                 bLog = TRUE;
314         iThreads = 4;
315         size = sizeof(szTmp);
316         if ( RegQueryValueEx(hKey, "NumberOfDelivery-
 Threads", 0, &type, szTmp, &size) == ERROR_SUCCESS )
317             iThreads = atoi(szTmp);
318         if ( iThreads == 0 )
319             iThreads = 4;
320
321         iMaxWareHouse = 500;
322         size = sizeof(szTmp);
323         if ( RegQueryValueEx(hKey, "MaximumWare-
 houses", 0, &type, szTmp, &size) == ERROR_SUCCESS )
324             iMaxWareHouse = atoi(szTmp);
325         if ( iMaxWareHouse == 0 )
326             iMaxWareHouse = 500;
327
328         iDelayMs = 500;
329         size = sizeof(szTmp);
330         if ( RegQueryValueEx(hKey, "BackoffDelay", 0,
 &type, szTmp, &size) == ERROR_SUCCESS )
331             iDelayMs = atoi(szTmp);
```

```
332         if ( iDelayMs == 0 )
333             iDelayMs = 500;
334
335         iDeadlockRetry = 3;
336         size = sizeof(szTmp);
337        if ( RegQueryValueEx(hKey, "DeadlockRetry", 0,
 &type, szTmp, &size) == ERROR_SUCCESS )
338             iDeadlockRetry = atoi(szTmp);
339         if ( !iDeadlockRetry )
340             iDeadlockRetry = 3;
341
342         iMaxConnections = 25;
343         size = sizeof(szTmp);
344         if ( RegQueryValueEx(hKey, "MaxConnections",
 0, &type, szTmp, &size) == ERROR_SUCCESS )
345             iMaxConnections = atoi(szTmp);
346         if ( !iMaxConnections )
347             iMaxConnections = 25;
348
349         bConnectionPooling = FALSE;
350         size = sizeof(szTmp);
351         if ( RegQueryValueEx(hKey, "ConnectionPool-
 ing", 0, &type, szTmp, &size) == ERROR_SUCCESS )
352             if ( !stricmp(szTmp, "ON") )
353                 bConnectionPooling = TRUE;
354
355         iConnectDelay = 500;
356         size = sizeof(szTmp);
357         if ( RegQueryValueEx(hKey, "ConnectionPoolRe-
 tryTime", 0, &type, szTmp, &size) == ERROR_SUCCESS )
358             iConnectDelay = atoi(szTmp);
359         if ( !iConnectDelay )
360             iConnectDelay = 500;
361
362         strcpy(szVersion, "DBLIB");
363         size = sizeof(szTmp);
364         if ( RegQueryValueEx(hKey, "LastInstalledVer-
 sion", 0, &type, szTmp, &size) == ERROR_SUCCESS )
365             strcpy(szVersion, szTmp);
366
367         if ( strcmp(szVersion, "DBLIB") != 0 &&
 strcmp(szVersion, "ODBC") != 0 )
368             strcpy(szVersion, "DBLIB");
369
370         RegCloseKey(hKey);
371
372         if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SYS-
 TEM\\CurrentControlSet\\Services\\Inetinfo\\Parame-
 ters", 0, KEY_READ, &hKey) == ERROR_SUCCESS )
373         {
374             iPoolThreadLimit =  iMaxPhysicalMemory *
 2;
375             size = sizeof(iPoolThreadLimit);
376             if ( RegQueryValueEx(hKey, "PoolThread-
 Limit", 0, &type, (char *)&iPoolThreadLimit, &size) ==
 ERROR_SUCCESS )
377             if ( !iPoolThreadLimit )
378               iPoolThreadLimit =  iMaxPhysicalMemory
 * 2;
379
380             iThreadTimeout = 86400;
381             size = sizeof(iThreadTimeout);
382             if ( RegQueryValueEx(hKey, "ThreadTime-
 out", 0, &type, (char *)&iThreadTimeout, &size) ==
 ERROR_SUCCESS )
383                 if ( !iThreadTimeout )
384                     iThreadTimeout = 86400;
385
386             iListenBackLog = 15;
387             size = sizeof(iListenBackLog);
388             if ( RegQueryValueEx(hKey, "ListenBack-
 Log", 0, &type, (char *)&iListenBackLog, &size) ==
 ERROR_SUCCESS )
389                 if ( !iListenBackLog )
390                     iListenBackLog = 15;
391         }
392
393         RegCloseKey(hKey);
394
395         if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SYS-
 TEM\\CurrentControlSet\\Services\\W3SVC\\Parameters",
 0, KEY_READ, &hKey) == ERROR_SUCCESS )
396         {
397             iAcceptExOutstanding = 40;
398             size = sizeof(iAcceptExOutstanding);
399             if ( RegQueryValueEx(hKey, "AcceptExOut-
 standing", 0, &type, (char *)&iAcceptExOutstanding,
 &size) == ERROR_SUCCESS )
400                 if ( !iAcceptExOutstanding )
401                     iAcceptExOutstanding = 40;
402         }
403
404         RegCloseKey(hKey);
405     }
406
407     return;
408 }
409
410 static void WriteRegistrySettings(char *szDllPath)
411 {
412     HKEY    hKey;
413     DWORD   dwDisposition;
414     char    szTmp[256];
415     char    *ptr;
416     int     iRc;
417
418     if ( RegCreateKeyEx(HKEY_LOCAL_MACHINE, "SOFT-
 WARE\\Microsoft\\TPCC", 0, NULL,
 REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &hKey,
 &dwDisposition) == ERROR_SUCCESS )
419     {
420         strcpy(szTmp, szDllPath);
421         ptr = strstr(szTmp, "tpcc");
422         if ( ptr )
423             *ptr = 0;
424
425        RegSetValueEx(hKey, "PATH", 0, REG_SZ, szTmp,
 strlen(szTmp));
426
427         if ( bLog )
428             RegSetValueEx(hKey, "LOG", 0, REG_SZ,
 "ON", 2);
429         else
430             RegSetValueEx(hKey, "LOG", 0, REG_SZ,
 "OFF", 3);
431
432         itoa(iThreads, szTmp, 10);
433       RegSetValueEx(hKey, "NumberOfDeliveryThreads",
 0, REG_SZ, szTmp, strlen(szTmp));
434
435         itoa(iMaxWareHouse, szTmp, 10);
436         RegSetValueEx(hKey, "MaximumWarehouses", 0,
 REG_SZ, szTmp, strlen(szTmp));
437
438         itoa(iDelayMs, szTmp, 10);
439       RegSetValueEx(hKey, "BackoffDelay", 0, REG_SZ,
 szTmp, strlen(szTmp));
440
441         itoa(iDeadlockRetry, szTmp, 10);
442         RegSetValueEx(hKey, "DeadlockRetry", 0,
 REG_SZ, szTmp, strlen(szTmp));
443
444         itoa(iMaxConnections, szTmp, 10);
445         RegSetValueEx(hKey, "MaxConnections", 0,
 REG_SZ, szTmp, strlen(szTmp));
446
447         itoa(iMaxConnections, szTmp, 10);
448         RegSetValueEx(hKey, "MaxConnections", 0,
 REG_SZ, szTmp, strlen(szTmp));
449
450         if ( bConnectionPooling )
451             RegSetValueEx(hKey, "ConnectionPooling",
 0, REG_SZ, "ON", 2);
452         else
```

```
453             RegSetValueEx(hKey, "ConnectionPooling",
 0, REG_SZ, "OFF", 3);
454
455          itoa(iConnectDelay, szTmp, 10);
456        RegSetValueEx(hKey, "ConnectionPoolRetryTime",
 0, REG_SZ, szTmp, strlen(szTmp));
457
458        RegSetValueEx(hKey, "LastInstalledVersion", 0,
 REG_SZ, szVersion, strlen(szVersion));
459
460          RegFlushKey(hKey);
461
462          RegCloseKey(hKey);
463      }
464
465      if ( (iRc=RegCreateKeyEx(HKEY_LOCAL_MACHINE,
 "SYSTEM\\CurrentControlSet\\Services\\Inetinfo\\Parame-
 ters", 0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
 NULL, &hKey, &dwDisposition)) == ERROR_SUCCESS )
466      {
467          RegSetValueEx(hKey, "PoolThreadLimit", 0,
 REG_DWORD, (char *)&iPoolThreadLimit, sizeof(iPoolTh-
 readLimit));
468          RegSetValueEx(hKey, "ThreadTimeout", 0,
 REG_DWORD, (char *)&iThreadTimeout, sizeof(iThreadTime-
 out));
469          RegSetValueEx(hKey, "ListenBackLog", 0,
 REG_DWORD, (char *)&iListenBackLog, sizeof(iListenBack-
 Log));
470
471          RegFlushKey(hKey);
472          RegCloseKey(hKey);
473      }
474
475      if ( (iRc=RegCreateKeyEx(HKEY_LOCAL_MACHINE,
 "SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parame-
 ters", 0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
 NULL, &hKey, &dwDisposition)) == ERROR_SUCCESS )
476      {
477         RegSetValueEx(hKey, "AcceptExOutstanding", 0,
 REG_DWORD, (char *)&iAcceptExOutstanding, sizeof(iAccep-
 tExOutstanding));
478
479          RegFlushKey(hKey);
480          RegCloseKey(hKey);
481      }
482
483      return;
484  }
485
486  BOOL CALLBACK CopyDlgProc(HWND hwnd, UINT uMsg,
 WPARAM wParam, LPARAM lParam)
487  {
488      if ( uMsg == WM_INITDIALOG )
489      {
490          SendDlgItemMessage(hwnd, IDC_PROGRESS1,
 PBM_SETRANGE, 0, MAKELPARAM(0, 8));
491          SendDlgItemMessage(hwnd, IDC_PROGRESS1,
 PBM_SETSTEP, (WPARAM)1, 0);
492          return TRUE;
493      }
494      return FALSE;
495  }
496
497  static int CopyFiles(HWND hDlg, char *szDllPath)
498  {
499      HGLOBAL         hDLL;
500      HGLOBAL         hExe;
501      HRSRC           hResInfo;
502      BYTE            *pSrc;
503      HANDLE          hFile;
504      DWORD           dwSize;
505      DWORD           d;
506      char            szTmp[256];
507      char            *ptr;
508      BOOL            bSvcRunning;
509
510
511      bSvcRunning = CheckWWWebService();
512      if ( bSvcRunning )
513      {
514         SetDlgItemText(hDlg, IDC_STATUS, "Stopping Web
 Service.");
515          SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
516          UpdateDialog(hDlg);
517
518          StopWWWebService();
519          SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
520          UpdateDialog(hDlg);
521      }
522
523      if ( iDllType == IDC_DBLIB )
524          hResInfo = FindResource(hInst, MAKEINTRE-
 SOURCE(IDR_TPCCDLL1), "TPCCDLL");
525      else // iDllType == IDC_ODBC
526          hResInfo = FindResource(hInst, MAKEINTRE-
 SOURCE(IDR_TPCCDLL2), "TPCCDLL");
527
528      SetDlgItemText(hDlg, IDC_STATUS, "Copying
 Files...");
529      SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
530      UpdateDialog(hDlg);
531
532      dwSize = SizeofResource(hInst, hResInfo);
533      hDLL = LoadResource(hInst, hResInfo );
534      pSrc = (BYTE *)LockResource(hDLL);
535      remove(szDllPath);
536
537      if ( !(hFile = CreateFile(szDllPath,
 GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
 FILE_ATTRIBUTE_NORMAL, NULL)) )
538          return 0;
539
540      if ( !WriteFile(hFile, pSrc, dwSize, &d, NULL) )
541          return 0;
542
543      CloseHandle(hFile);
544
545      UnlockResource(hDLL);
546      FreeResource(hDLL);
547
548      SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
549      UpdateDialog(hDlg);
550
551      if ( iDllType == IDC_DBLIB )
552          hResInfo = FindResource(hInst, MAKEINTRE-
 SOURCE(IDR_DELIVERY1), "DELIVERY");
553      else
554          hResInfo = FindResource(hInst, MAKEINTRE-
 SOURCE(IDR_DELIVERY2), "DELIVERY");
555
556      dwSize = SizeofResource(hInst, hResInfo);
557      hExe = LoadResource(hInst, hResInfo );
558      pSrc = (BYTE *)LockResource(hExe);
559
560      strcpy(szTmp, szDllPath);
561      ptr = strstr(szTmp, "tpcc");
562      if ( ptr )
563          *ptr = 0;
564      strcat(szTmp, "delisrv.exe");
565
566      remove(szTmp);
567
568     if ( !(hFile = CreateFile(szTmp, GENERIC_WRITE, 0,
 NULL, CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL)) )
569          return 0;
570
571      if ( !WriteFile(hFile, pSrc, dwSize, &d, NULL) )
572          return 0;
573
```

```
574     CloseHandle(hFile);
575
576     UnlockResource(hExe);
577     FreeResource(hExe);
578
579     SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
580     UpdateDialog(hDlg);
581
582     //if we stopped service restart it.
583     if ( bSvcRunning )
584     {
585        SetDlgItemText(hDlg, IDC_STATUS, "Starting Web
 Service.");
586         SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
587         UpdateDialog(hDlg);
588         StartWWWebService();
589     }
590
591     SendDlgItemMessage(hDlg, IDC_PROGRESS1,
 PBM_STEPIT, 0, 0);
592     UpdateDialog(hDlg);
593
594     return 1;
595 }
596
597 static BOOL GetInstallPath(char *szDllPath)
598 {
599     HKEY    hKey;
600     BYTE    szTmp[256];
601     char    szKey[256];
602     DWORD   size;
603     DWORD   sv;
604     BOOL    bRc;
605     int     len;
606     char    *ptr;
607
608     szDllPath[0] =  0;
609     bRc = TRUE;
610     if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SYS-
 TEM\\CurrentControlSet\\Services\\W3SVC\\Parame-
 ters\\Virtual Roots", 0, KEY_ALL_ACCESS, &hKey) ==
 ERROR_SUCCESS )
611     {
612         sv = sizeof(szKey);
613         size = sizeof(szTmp);
614
615         if ( RegEnumValue(hKey, 0, szKey, &sv, NULL,
 NULL, szTmp, &size) == ERROR_SUCCESS )
616         {
617             strcpy(szDllPath, szTmp);
618             bRc = FALSE;
619         }
620         RegCloseKey(hKey);
621     }
622     if ( (ptr = strchr(szDllPath, ',')) )
623         *ptr = 0;
624
625     len = strlen(szDllPath);
626     if ( szDllPath[len-1] != '\\' )
627     {
628         szDllPath[len] = '\\';
629         szDllPath[len+1] = 0;
630     }
631     strcat(szDllPath, "tpcc.dll");
632
633     return bRc;
634 }
635
636 static void GetVersionInfo(char *szDLLPath, char
 *szExePath)
637 {
638     DWORD               d;
639     DWORD               dwSize;
640     DWORD               dwBytes;
641     char                *ptr;
642     VS_FIXEDFILEINFO    *vs;
643
644     versionDllMS = 0;
645     versionDllLS = 0;
646     if ( _access(szDLLPath, 00) == 0 )
647     {
648         dwSize = GetFileVersionInfoSize(szDLLPath,
 &d);
649         if ( dwSize )
650         {
651             ptr = (char *)malloc(dwSize);
652             GetFileVersionInfo(szDLLPath, 0, dwSize,
 ptr);
653             VerQueryValue(ptr, "\\",&vs, &dwBytes);
654             versionDllMS = vs->dwProductVersionMS;
655             versionDllLS = vs->dwProductVersionLS;
656             free(ptr);
657         }
658     }
659
660     versionExeMS = 0x7FFF;
661     versionExeLS = 0x7FFF;
662     dwSize = GetFileVersionInfoSize(szExePath, &d);
663     if ( dwSize )
664     {
665         ptr = (char *)malloc(dwSize);
666        GetFileVersionInfo(szExePath, 0, dwSize, ptr);
667         VerQueryValue(ptr, "\\",&vs, &dwBytes);
668
669         versionExeMS = vs->dwProductVersionMS;
670        versionExeLS = LOWORD(vs->dwProductVersionLS);
671        versionExeMM = HIWORD(vs->dwProductVersionLS);
672         free(ptr);
673     }
674     return;
675 }
676
677 static BOOL CheckWWWebService(void)
678 {
679     SC_HANDLE       schSCManager;
680     SC_HANDLE       schService;
681     SERVICE_STATUS  ssStatus;
682
683     schSCManager = OpenSCManager(NULL, NULL,
 SC_MANAGER_ALL_ACCESS);
684     schService = OpenService(schSCManager,
 TEXT("W3SVC"), SERVICE_ALL_ACCESS);
685     if (schService == NULL)
686         return FALSE;
687
688     if (! QueryServiceStatus(schService, &ssStatus) )
689         goto ServiceNotRunning;
690
691     if ( !ControlService(schService,
 SERVICE_CONTROL_STOP, &ssStatus) )
692         goto ServiceNotRunning;
693     //start Service pending, Check the status until
 the service is running.
694     if (! QueryServiceStatus(schService, &ssStatus) )
695         goto ServiceNotRunning;
696
697     CloseServiceHandle(schService);
698     return TRUE;
699
700 ServiceNotRunning:
701
702     CloseServiceHandle(schService);
703     return FALSE;
704 }
705
706 static BOOL StartWWWebService(void)
707 {
708     SC_HANDLE       schSCManager;
709     SC_HANDLE       schService;
710     SERVICE_STATUS  ssStatus;
711     DWORD           dwOldCheckPoint;
712
```

```
713     schSCManager = OpenSCManager(NULL, NULL,
 SC_MANAGER_ALL_ACCESS);
714     schService = OpenService(schSCManager,
 TEXT("W3SVC"), SERVICE_ALL_ACCESS);
715     if (schService == NULL)
716         return FALSE;
717
718     if (! StartService(schService, 0, NULL) )
719         goto StartWWWebErr;
720     //start Service pending, Check the status until
 the service is running.
721     if (! QueryServiceStatus(schService, &ssStatus) )
722         goto StartWWWebErr;
723     while( ssStatus.dwCurrentState !=
 SERVICE_RUNNING)
724     {
725
726         dwOldCheckPoint = ssStatus.dwCheck-
 Point;            //Save the current checkpoint.
727         Sleep(ssStatus.dwWait-
 Hint);                        //Wait for the specified
 interval.
728         if ( !QueryServiceStatus(schService, &ssSta-
 tus) )   //Check the status again.
729             break;
730      if (dwOldCheckPoint >= ssStatus.dwCheckPoint)
 //Break if the checkpoint has not been incremented.
731             break;
732     }
733
734     if (ssStatus.dwCurrentState == SERVICE_RUNNING)
735         goto StartWWWebErr;
736
737     CloseServiceHandle(schService);
738     return TRUE;
739
740 StartWWWebErr:
741     CloseServiceHandle(schService);
742     return FALSE;
743 }
744
745 static BOOL StopWWWebService(void)
746 {
747     SC_HANDLE       schSCManager;
748     SC_HANDLE       schService;
749     SERVICE_STATUS  ssStatus;
750     DWORD           dwOldCheckPoint;
751
752     schSCManager = OpenSCManager(NULL, NULL,
 SC_MANAGER_ALL_ACCESS);
753     schService = OpenService(schSCManager,
 TEXT("W3SVC"), SERVICE_ALL_ACCESS);
754     if (schService == NULL)
755         return FALSE;
756
757     if (! QueryServiceStatus(schService, &ssStatus) )
758         goto StopWWWebErr;
759
760     if ( !ControlService(schService,
 SERVICE_CONTROL_STOP, &ssStatus) )
761         goto StopWWWebErr;
762     //start Service pending, Check the status until
 the service is running.
763     if (! QueryServiceStatus(schService, &ssStatus) )
764         goto StopWWWebErr;
765     while( ssStatus.dwCurrentState ==
 SERVICE_RUNNING)
766     {
767
768         dwOldCheckPoint = ssStatus.dwCheck-
 Point;            //Save the current checkpoint.
769         Sleep(ssStatus.dwWait-
 Hint);                        //Wait for the specified
 interval.
770         if ( !QueryServiceStatus(schService, &ssSta-
 tus) )   //Check the status again.
771             break;
772      if (dwOldCheckPoint >= ssStatus.dwCheckPoint)
 //Break if the checkpoint has not been incremented.
773             break;
774     }
775
776     if (ssStatus.dwCurrentState == SERVICE_RUNNING)
777         goto StopWWWebErr;
778
779     CloseServiceHandle(schService);
780     return TRUE;
781
782 StopWWWebErr:
783     CloseServiceHandle(schService);
784     return FALSE;
785 }
786
787 static void UpdateDialog(HWND hDlg)
788 {
789     MSG msg;
790
791     UpdateWindow(hDlg);
792     while( PeekMessage(&msg, hDlg, 0, 0, PM_REMOVE) )
793     {
794         TranslateMessage(&msg);
795         DispatchMessage(&msg);
796     }
797     Sleep(250);
798     return;
799 }
```

install.h

```
1  //{{NO_DEPENDENCIES}}
2  // Microsoft Developer Studio generated include file.
3  // Used by install.rc
4  //
5
6  #define IDD_DIALOG1                     101
7  #define IDI_ICON1                       102
8  #define IDR_TPCCDLL1                    103
9  #define IDR_TPCCDLL2                    104
10 #define IDD_DIALOG2                     105
11 #define IDI_ICON2                       106
12 #define IDR_DELIVERY1                   107
13 #define IDD_DIALOG3                     108
14 #define IDR_DELIVERY2                   109
15
16 #define BN_LOG                          1001
17 #define ED_KEEP                         1002
18 #define ED_THREADS                      1003
19 #define ED_THREADS2                     1004
20 #define ED_MAXWARE                      1006
21 #define IDC_PATH                        1007
22 #define IDC_VERSION                     1009
23 #define IDC_RESULTS                     1010
24 #define IDC_PROGRESS1                   1011
25 #define IDC_STATUS                      1012
26 #define IDC_BUTTON1                     1013
27 #define ED_MAXCONNECTION                1014
28 #define ED_IIS_MAX_THEAD_POOL_LIMIT     1015
29 #define ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE   1017
30 #define ED_IIS_THREAD_TIMEOUT           1018
31 #define ED_IIS_LISTEN_BACKLOG           1019
32 #define IDC_DBLIB                       1021
33 #define IDC_ODBC                        1022
34 #define IDC_CONNECT_POOL                1023
35 #define ED_USER_CONNECT_DELAY_TIME      1024
36
37 // Next default values for new objects
38 //
39 #ifdef APSTUDIO_INVOKED
40 #ifndef APSTUDIO_READONLY_SYMBOLS
41 #define _APS_NEXT_RESOURCE_VALUE        111
42 #define _APS_NEXT_COMMAND_VALUE         40001
43 #define _APS_NEXT_CONTROL_VALUE         1022
44 #define _APS_NEXT_SYMED_VALUE           101
45 #endif
```

```
46  #endif
```

install.rc

```
1    //Microsoft Developer Studio generated resource
script.
2    //
3    #include "install.h"
4
5    #define APSTUDIO_READONLY_SYMBOLS
6    //////////////////////
7    //
8    // Generated from the TEXTINCLUDE 2 resource.
9    //
10   #include "afxres.h"
11
12   //////////////////////
13   #undef APSTUDIO_READONLY_SYMBOLS
14
15   //////////////////////
16   // English (U.S.) resources
17
18   #if !defined(AFX_RESOURCE_DLL) ||
defined(AFX_TARG_ENU)
19   #ifdef _WIN32
20   LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
21   #pragma code_page(1252)
22   #endif //_WIN32
23
24   //////////////////////
25   //
26   // Dialog
27   //
28
29   IDD_DIALOG1 DIALOGEX 0, 0, 218, 250
30   STYLE DS_MODALFRAME | DS_CENTER | WS_MINIMIZEBOX |
WS_POPUP | WS_CAPTION |
31       WS_SYSMENU
32   CAPTION "TPC-C Web Client Installation Utility"
33   FONT 8, "MS Sans Serif"
34   BEGIN
35       EDITTEXT
ED_MAXWARE,183,37,21,12,ES_NUMBER,WS_EX_RTLREADING
36       CONTROL         "",BN_LOG,"But-
ton",BS_AUTOCHECKBOX | BS_LEFTTEXT |
37                       BS_LEFT | BS_VCENTER |
WS_TABSTOP,189,51,15,13,
38                       WS_EX_STATICEDGE
39       EDITTEXT
ED_THREADS,189,65,15,12,ES_NUMBER,WS_EX_RTLREADING
40       EDITTEXT
ED_MAXCONNECTION,170,79,34,12,ES_NUMBER,WS_EX_RTLREADING
41       EDITTEXT
ED_IIS_MAX_THEAD_POOL_LIMIT,170,93,34,12,ES_NUMBER,
42                       WS_EX_RTLREADING
43       EDITTEXT
ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE,170,107,34,12,
44                       ES_NUMBER,WS_EX_RTLREADING
45       EDITTEXT
ED_IIS_THREAD_TIMEOUT,170,121,34,12,ES_NUMBER,
46                       WS_EX_RTLREADING
47       EDITTEXT
ED_IIS_LISTEN_BACKLOG,170,135,34,12,ES_NUMBER,
48                       WS_EX_RTLREADING
49       CONTROL         "DBLIB",IDC_DBLIB,"But-
ton",BS_AUTORADIOBUTTON |
50                       WS_TABSTOP,162,152,39,12
51       CONTROL         "ODBC",IDC_ODBC,"But-
ton",BS_AUTORADIOBUTTON | WS_TABSTOP,
52                       162,167,39,12
53       CONTROL         "",IDC_CONNECT_POOL,"But-
ton",BS_AUTOCHECKBOX |
54                       BS_LEFTTEXT | BS_LEFT |
BS_VCENTER | WS_TABSTOP,189,190,
55                       15,13,WS_EX_STATICEDGE
56       EDITTEXT
ED_USER_CONNECT_DELAY_TIME,170,206,34,12,ES_NUMBER,
57                       WS_EX_RTLREADING
58       DEFPUSHBUTTON   "OK",IDOK,51,229,50,14
59       PUSHBUTTON      "Cancel",IDCANCEL,117,229,50,14
60       EDITTEXT
IDC_PATH,42,22,162,13,ES_AUTOHSCROLL | ES_READONLY
61       LTEXT           "Max Number of Ware-
houses:",IDC_STATIC,42,37,115,12,
62                       SS_SUNKEN
63       LTEXT           "Write HTML To Log
file:",IDC_STATIC,42,51,115,12,
64                       SS_SUNKEN
65       LTEXT           "Number of Delivery
Threads:",IDC_STATIC,42,65,115,12,
66                       SS_SUNKEN
67       LTEXT           "Max Number of Connec-
tions:",IDC_STATIC,42,79,115,12,
68                       SS_SUNKEN
69       CTEXT           "Version
1.00.001",IDC_VERSION,42,6,162,14,SS_SUNKEN |
70                       WS_BORDER,WS_EX_CLIENTEDGE
71       ICON
IDI_ICON1,IDC_STATIC,9,6,21,20,0,WS_EX_CLIENTEDGE
72       LTEXT           "IIS Max Thread Pool
Limit:",IDC_STATIC,42,93,115,12,
73                       SS_SUNKEN
74       LTEXT           "Web Service Backlog Queue
Size:",IDC_STATIC,42,107,115,
75                       12,SS_SUNKEN
76       LTEXT           "IIS Thread Time-
out:",IDC_STATIC,42,121,115,12,SS_SUNKEN
77       LTEXT           "IIS Listen Back-
log:",IDC_STATIC,42,135,115,12,SS_SUNKEN
78       LTEXT           "Database Inter-
face:",IDC_STATIC,42,159,115,12,SS_SUNKEN
79       GROUPBOX        "",IDC_STATIC,160,146,44,38
80       LTEXT           "Use ODBC Connection Pool-
ing:",IDC_STATIC,42,190,115,12,
81                       SS_SUNKEN
82       LTEXT           "Connection Pool Retry
Delay:",IDC_STATIC,42,206,
83                       115,12,SS_SUNKEN
84   END
85
86   IDD_DIALOG2 DIALOGEX 0, 0, 117, 62
87   STYLE DS_SETFOREGROUND | DS_3DLOOK | DS_CENTER |
WS_POPUP | WS_BORDER
88   EXSTYLE WS_EX_STATICEDGE
89   FONT 12, "MS Sans Serif", 0, 0, 0x1
90   BEGIN
91       DEFPUSHBUTTON   "OK",IDOK,33,45,50,9
92       CTEXT           "HTML TPC-C Installation Suc-
cessfull",IDC_RESULTS,7,22,
93                       102,18,0,WS_EX_CLIENTEDGE
94       ICON
IDI_ICON2,IDC_STATIC,50,7,18,20,SS_REALSIZEIMAGE,
95                       WS_EX_TRANSPARENT
96   END
97
98   IDD_DIALOG3 DIALOG DISCARDABLE  0, 0, 91, 40
99   STYLE DS_SYSMODAL | DS_MODALFRAME | DS_3DLOOK |
DS_CENTER | WS_CAPTION
100  CAPTION "Installing TPC-C Web Client"
101  FONT 12, "Arial Black"
102  BEGIN
103      CONTROL
 "Progress1",IDC_PROGRESS1,"msctls_progress32",WS_BORDER,
104                      7,20,77,13
105      CTEXT
 "Static",IDC_STATUS,7,7,77,12,SS_SUNKEN
106  END
107
108
109  //////////////////////
110  //
111  // DESIGNINFO
112  //
113
```

```
114    #ifdef APSTUDIO_INVOKED
115    GUIDELINES DESIGNINFO DISCARDABLE
116    BEGIN
117        IDD_DIALOG1, DIALOG
118        BEGIN
119            LEFTMARGIN, 9
120            RIGHTMARGIN, 204
121            TOPMARGIN, 6
122            BOTTOMMARGIN, 232
123        END
124
125        IDD_DIALOG2, DIALOG
126        BEGIN
127            LEFTMARGIN, 7
128            RIGHTMARGIN, 109
129            TOPMARGIN, 7
130            BOTTOMMARGIN, 54
131        END
132
133        IDD_DIALOG3, DIALOG
134        BEGIN
135            LEFTMARGIN, 7
136            RIGHTMARGIN, 84
137            TOPMARGIN, 7
138            BOTTOMMARGIN, 33
139        END
140    END
141    #endif    // APSTUDIO_INVOKED
142
143
144    #ifdef APSTUDIO_INVOKED
145    ////////////////////
146    //
147    // TEXTINCLUDE
148    //
149
150    1 TEXTINCLUDE DISCARDABLE
151    BEGIN
152        "resource.h\0"
153    END
154
155    2 TEXTINCLUDE DISCARDABLE
156    BEGIN
157        "#include ""afxres.h""\r\n"
158        "\0"
159    END
160
161    3 TEXTINCLUDE DISCARDABLE
162    BEGIN
163        "\r\n"
164        "\0"
165    END
166
167    #endif    // APSTUDIO_INVOKED
168
169
170    ////////////////////
171    //
172    // Icon
173    //
174
175    // Icon with lowest ID value placed first to ensure
 application icon
176    // remains consistent on all systems.
177    IDI_ICON1               ICON    DISCARDABLE
 "icon1.ico"
178    IDI_ICON2               ICON    DISCARDABLE
 "icon2.ico"
179
180    ////////////////////
181    //
182    // TPCCDLL
183    //
184
185    IDR_TPCCDLL1            TPCCDLL DISCARDABLE
 "tpcc1.dll"
186    IDR_TPCCDLL2            TPCCDLL DISCARDABLE
 "tpcc2.dll"
187
188    #ifndef _MAC
189    ////////////////////
190    //
191    // Version
192    //
193
194    VS_VERSION_INFO VERSIONINFO
195     FILEVERSION 0,4,0,0
196     PRODUCTVERSION 0,4,0,0
197     FILEFLAGSMASK 0x3fL
198    #ifdef _DEBUG
199     FILEFLAGS 0x1L
200    #else
201     FILEFLAGS 0x0L
202    #endif
203     FILEOS 0x40004L
204     FILETYPE 0x1L
205     FILESUBTYPE 0x0L
206    BEGIN
207        BLOCK "StringFileInfo"
208        BEGIN
209            BLOCK "040904b0"
210            BEGIN
211                VALUE "CompanyName", "Microsoft\0"
212                VALUE "FileDescription", "install\0"
213                VALUE "FileVersion", "0, 4, 0, 0\0"
214                VALUE "InternalName", "install\0"
215                VALUE "LegalCopyright", "Copyright ©
 1996\0"
216                VALUE "OriginalFilename",
 "install.exe\0"
217                VALUE "ProductName", "Microsoft
 install\0"
218                VALUE "ProductVersion", "0, 4, 0, 0\0"
219            END
220        END
221        BLOCK "VarFileInfo"
222        BEGIN
223            VALUE "Translation", 0x409, 1200
224        END
225    END
226
227    #endif    // !_MAC
228
229
230    ////////////////////
231    //
232    // DELIVERY
233    //
234
235    IDR_DELIVERY1           DELIVERY DISCARDABLE
 "delisrv1.exe"
236    IDR_DELIVERY2           DELIVERY DISCARDABLE
 "delisrv2.exe"
237    #endif    // English (U.S.) resources
238    ////////////////////
239
240
241
242    #ifndef APSTUDIO_INVOKED
243    ////////////////////
244    //
245    // Generated from the TEXTINCLUDE 3 resource.
246    //
247
248
249    ////////////////////
250    #endif    // not APSTUDIO_INVOKED
251
```

pipe_routines.c

```
  1  #include <windows.h>
  2  #include <stdio.h>
  3  #include "pipe_routines.h"
```

```
  4  #include "trans.h"
  5  #include "tpcc.h"
  6  #include "tux.h"
  7
  8  const int MAXRETRIES=600;
  9  const int SLEEP_TIME=100;
 10
 11  const char *SERVER_PIPE_PATH =
"\\\\.\\pipe\\tpcc_pipe.%d";
 12  const char *CLIENT_PIPE_PATH =
"\\\\.\\pipe\\tpcc_pipe.%d";
 13
 14  HANDLE
 15  OpenServerPipe(int PipeNumber, int TimeOut)
 16  {
 17      HANDLE hPipe, hEvent;
 18      OVERLAPPED overlapped;
 19      BOOL bSuccess;
 20      char PipeName[_MAX_PATH];
 21      SECURITY_ATTRIBUTES     sa;
 22      PSECURITY_DESCRIPTOR    pSD;
 23
 24      _snprintf(PipeName, sizeof(PipeName),
SERVER_PIPE_PATH, PipeNumber);
 25
 26  #ifdef _DEBUG
 27      fprintf(stderr, "opening server pipe %s\n", Pipe-
Name);
 28  #endif
 29
 30      hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
 31      if (hEvent == INVALID_HANDLE_VALUE)
 32      {
 33          fprintf(stderr, "OpenServerPipe(%d): Unable
to create event handle\n", PipeNumber);
 34          return INVALID_HANDLE_VALUE;
 35      }
 36
 37      // create a security descriptor that allows any-
one to access the pipe...
 38      pSD = (PSECURITY_DESCRIPTOR)malloc(
SECURITY_DESCRIPTOR_MIN_LENGTH );
 39      InitializeSecurityDescriptor(pSD,
SECURITY_DESCRIPTOR_REVISION);
 40      SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL,
FALSE);
 41      sa.nLength              = sizeof(sa);
 42      sa.lpSecurityDescriptor = pSD;
 43      sa.bInheritHandle       = TRUE;
 44
 45      hPipe = CreateNamedPipe(PipeName,
 46                  PIPE_ACCESS_DUPLEX |
FILE_FLAG_OVERLAPPED,
 47                  PIPE_TYPE_MESSAGE |
PIPE_READMODE_MESSAGE,
 48                  1,
 49                  sizeof(TUX_MSG),
 50                  sizeof(TUX_MSG),
 51                  0,
 52                  &sa);
 53
 54      if (hPipe == INVALID_HANDLE_VALUE)
 55      {
 56          fprintf(stderr, "OpenServerPipe(%d): Create-
HamedPipe failed with error %d\n", PipeNumber, GetLastEr-
ror());
 57          CloseHandle(hEvent);
 58          return INVALID_HANDLE_VALUE;
 59      }
 60
 61      overlapped.hEvent = hEvent;
 62      ConnectNamedPipe(hPipe, &overlapped);
 63
 64      bSuccess = TRUE; // wish for the best
 65      switch (GetLastError())
 66      {
 67          case ERROR_PIPE_CONNECTED:
 68              // someone had connected between the cre-
ate at the connect call - no biggie
 69              break;
 70          case ERROR_IO_PENDING:
 71              // no one was waiting for us. Set a time-
out and wait for them to
 72              // connect
 73              switch(WaitForSingleObject(hEvent, Time-
Out))
 74              {
 75                  case WAIT_OBJECT_0:
 76                      // Someone connected within the
timeout period.  Continue processing
 77                      break;
 78                  case WAIT_TIMEOUT:
 79                      bSuccess = FALSE;
 80                      break;
 81                  default:
 82                      fprintf(stderr, "OpenServer-
Pipe(%d): waitforsingleobject failed, error=%d\n", PipeN-
umber, GetLastError());
 83                      bSuccess = FALSE;
 84                      break;
 85              }
 86              break;
 87          default:
 88              fprintf(stderr, "OpenServerPipe(%d): con-
nectnamedpipe failed, error=%d\n", PipeNumber, GetLastEr-
ror());
 89              bSuccess = FALSE;
 90              break;
 91      }
 92
 93      CloseHandle(hEvent);
 94      if (!bSuccess)
 95      {
 96          CloseHandle(hPipe);
 97          hPipe = INVALID_HANDLE_VALUE;
 98      }
 99
100      return hPipe;
101  }
102
103  HANDLE
104  OpenClientPipe(int ClientNumber)
105  {
106      char PipeName[_MAX_PATH];
107      HANDLE hPipe;
108      DWORD DesiredMode = PIPE_READMODE_MESSAGE;
109      int nRetries;
110
111      _snprintf(PipeName, sizeof(PipeName),
CLIENT_PIPE_PATH, ClientNumber);
112  #ifdef _DEBUG
113      fprintf(stderr, "OpenClientPipe begins for client
%d (%s)\n", ClientNumber, PipeName);
114  #endif
115
116      for(nRetries=0;nRetries<MAXRETRIES;nRetries++)
117      {
118          hPipe = CreateFile(PipeName,
119                  GENERIC_READ | GENERIC_WRITE,
120                  FILE_SHARE_READ | FILE_SHARE_WRITE,
121                  NULL,
122                  OPEN_EXISTING,
123                  FILE_ATTRIBUTE_NORMAL,
124                  0);
125
126          if (hPipe != INVALID_HANDLE_VALUE)
127              break;
128
129          switch(GetLastError())
130          {
131              case ERROR_FILE_NOT_FOUND:
132                      // give the server a chance
133  #ifdef _DEBUG
134                  fprintf(stderr, "sleeping\n");
```

```
#endif
                    Sleep(SLEEP_TIME);
                    break;
                default:
                    fprintf(stderr, "OpenClientPipe(%d):
error in create of %s.  Error = %d\n", ClientNumber, Pipe-
Name, GetLastError());
                    return INVALID_HANDLE_VALUE;
                    break;
            }
        }

        if (hPipe == INVALID_HANDLE_VALUE)
        {
            fprintf(stderr, "Client %d unable to open a
pipe after %d retries with %d ms wait time\n",
                ClientNumber, MAXRETRIES, SLEEP_TIME);

            return INVALID_HANDLE_VALUE;
        }

       if (!SetNamedPipeHandleState(hPipe, &DesiredMode,
NULL, NULL))
        {
            fprintf(stderr, "OpenClientPipe(%d), SetNam-
edPipeHandleStated faield in OpenclientPipe, error=%d\n",
ClientNumber, GetLastError());
            CloseHandle(hPipe);
            return INVALID_HANDLE_VALUE;
        }

        return hPipe;
    }

    BOOL
    ReadPipe(HANDLE hPipe, HANDLE hEvent, void *Buffer,
DWORD BufSize, DWORD *pnRead)
    {
        OVERLAPPED overlapped;

        memset(&overlapped, 0, sizeof(overlapped));
        overlapped.hEvent = hEvent;
        if (!ReadFile(hPipe, Buffer, BufSize, pnRead,
&overlapped))
        {
            switch(GetLastError())
            {
                case ERROR_IO_PENDING:
                    if (GetOverlappedResult(hPipe, &over-
lapped, pnRead, TRUE))
                        break;
                    if (GetLastError() !=
ERROR_BROKEN_PIPE)
                        fprintf(stderr, "ReadPipe: Read-
file failed, error=%d\n", GetLastError());
                    return FALSE;
                    break;
                case ERROR_BROKEN_PIPE:
                    return FALSE;
                    break;
                default:
                    fprintf(stderr, "ReadPipe: Readfile
failed, error=%d\n", GetLastError());
                    return FALSE;
                    break;
            }
        }

        if (*pnRead == BufSize)
        {
            DWORD BytesLeft;

            if (!PeekNamedPipe(hPipe, NULL, 0, 0, NULL,
&BytesLeft))
            {
                fprintf(stderr,"ReadPipe: PeekNamedPipe
failed, error=%d\n", GetLastError());
                return FALSE;
            }

            if (BytesLeft)
            {
                fprintf(stderr, "ReadPipe: buffer too
small.  Size was %d, left=%d\n",
                        BufSize, BytesLeft);
                return FALSE;
            }
        }

        return TRUE;
    }

    BOOL
    WritePipe(HANDLE hPipe, HANDLE hEvent, void *Buffer,
DWORD BytesToWrite, DWORD *pnWritten)
    {
        OVERLAPPED overlapped;

        memset(&overlapped, 0, sizeof(overlapped));
        overlapped.hEvent = hEvent;
        if (!WriteFile(hPipe, Buffer, BytesToWrite,
pnWritten, &overlapped))
        {
            switch(GetLastError())
            {
                case ERROR_IO_PENDING:
                    if (GetOverlappedResult(hPipe, &over-
lapped, pnWritten, TRUE))
                        break;
                    if (GetLastError() !=
ERROR_BROKEN_PIPE)
                        fprintf(stderr, "WritePipe: Write-
file failed, error=%d\n", GetLastError());
                    return FALSE;
                    break;
                case ERROR_BROKEN_PIPE:
                    return FALSE;
                    break;
                default:
                    fprintf(stderr, "WritePipe: Writefile
failed, error=%d\n", GetLastError());
                    return FALSE;
                    break;
            }
        }

        if (*pnWritten != BytesToWrite)
        {
            fprintf(stderr,"WritePipe: nWritten (%d) !=
BytesToWrite(%d)\n",
                    *pnWritten, BytesToWrite);
        }

        return TRUE;
    }
```

pipe_routines.h

```
    #ifndef PIPE_ROUTINES_H_INCLUDED
    #define PIPE_ROUTINES_H_INCLUDED

    HANDLE OpenServerPipe(int PipeNumber, int TimeOut);
    BOOL ReadPipe(HANDLE hPipe, HANDLE hEvent, void
   *Buffer, DWORD BufSize, DWORD *pnRead);
    HANDLE OpenClientPipe(int ClientNumber);
    BOOL WritePipe(HANDLE hPipe, HANDLE hEvent, void
   *Buffer, DWORD BufSize, DWORD *pnWritten);
    #endif
```

resource.h

```
    //{{NO_DEPENDENCIES}}
    // Microsoft Developer Studio generated include file.
```

```
 3  // Used by TPCC.rc
 4  //
 5
 6  // Next default values for new objects
 7  //
 8  #ifdef APSTUDIO_INVOKED
 9  #ifndef APSTUDIO_READONLY_SYMBOLS
10  #define _APS_NEXT_RESOURCE_VALUE        101
11  #define _APS_NEXT_COMMAND_VALUE         40001
12  #define _APS_NEXT_CONTROL_VALUE         1000
13  #define _APS_NEXT_SYMED_VALUE           101
14  #endif
15  #endif
```

samples.mak

```
 1  !IF "$(CFG)" == ""
 2  CFG=Debug
 3  !MESSAGE No configuration specified.  Defaulting to
Debug
 4  !ENDIF
 5
 6  !IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
 7  !MESSAGE Invalid configuration "$(CFG)" specified.
 8  !MESSAGE You can specify a configuration when running
NMAKE on this makefile
 9  !MESSAGE by defining the macro CFG on the command
line.  For example:
10  !MESSAGE
11  !MESSAGE NMAKE CFG="Debug"
12  !MESSAGE
13  !MESSAGE Possible choices for configuration are:
14  !MESSAGE
15  !MESSAGE "Release"
16  !MESSAGE "Debug"
17  !MESSAGE
18  !ERROR An invalid configuration is specified.
19  !ENDIF
20
21  SRCDIR          = .\Src
22  OBJDIR          = .\Objs
23  OUTDIR          = .\Bin
24
25  !IF "$(CFG)" != "Debug"
26  CDEBUG       =
27  DEBUG        =
28  FLAGS        = /D "WIN32" /D "_WINDOWS"
29  OPT          = /Ot
30  !ELSE
31  CDEBUG       = /Zi /Yd
32  FLAGS        = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
33  OPT          = /Od
34  !ENDIF
35
36  LIBS         = wininet.lib advapi32.lib
37  LFLAGS       = /link /PDB:$(OBJDIR)\$(*B).pdb /INCRE-
MENTAL:NO $(LIBS)
38  CFLAGS       = /nologo $(CDEBUG) $(FLAGS) $(OPT)
/Fd$(OBJDIR)\$(*B).pdb /Fo$(OBJDIR)\$(*B).obj /Fe$(OUT-
DIR)\$(*B).exe $(SRCDIR)\$(*B).c
39  CC           = cl $(CFLAGS) $(LFLAGS)
40
41  All:        $(OBJDIR)\. $(OUTDIR)\. $(OUTDIR)\webc-
nct.exe $(OUTDIR)\webtest.exe $(OUTDIR)\delirpt.exe
42
43  $(OBJDIR)\.:
44      if not exist $(OBJDIR) md $(OBJDIR)
45
46  $(OUTDIR)\.:
47      if not exist $(OUTDIR) md $(OUTDIR)
48
49  $(OUTDIR)\webcnct.exe:  $(SRCDIR)\webcnct.c
50      $(CC)
51
52  $(OUTDIR)\webtest.exe:  $(SRCDIR)\webtest.c
53      $(CC)
54
55  $(OUTDIR)\delirpt.exe:  $(SRCDIR)\delirpt.c
56      $(CC)
```

sqlroutines.c

```
1    #include <windows.h>
2    #include <stdio.h>
3
4    #include "util.h"
5    #include "trans.h"
6    #include "tpcc.h"
7    #include "error.h"
8    #include "sqlroutines.h"
9    #include "db.h"
10
11   int err_handler(DBPROCESS *dbproc, int severity,
int dberr, int oserr, char *dberrstr, char *oserrstr);
12   int msg_handler(DBPROCESS *dbproc, DBINT msgno, int
msgstate, int severity, char *msgtext);
13   BOOL SQLDetectDeadlock(DBPROCESS *dbproc);
14
15   static  CRITICAL_SECTION       ErrorLogCriticalSec-
tion;
16
17   BOOL SQLThreadAttach(void)
18   {
19       return TRUE;
20   }
21   BOOL SQLThreadDetach(void)
22   {
23       return TRUE;
24   }
25
26   BOOL
27   SQLInit(void)
28   {
29   #ifdef USE_ODBC
30       extern HENV henv;
31
32       if ( SQLAllocEnv(&henv) == SQL_ERROR )
33       {
34           MessageBox(NULL, "Error SQLAllocEnv()",
"Init", MB_OK | MB_ICONSTOP);
35           return FALSE;
36       }
37
38       #if (ODBCVER >= 0x0300)
39
40           if ( bConnectionPooling )
41           {
42
43               /* added to make sure we go into connec-
tion pooling mode */
44               Beep(100,500);
45               Beep(1000,500);
46
47               if ( SQLSetEnvAttr(henv,
SQL_ATTR_ODBC_VERSION, (PTR)SQL_OV_ODBC3, SQL_INTEGER)
== SQL_ERROR )
48               {
49                   MessageBox(NULL, "Error SQLSetEn-
vAttr() SQL_ATTR_ODBC_VERSION", "Init", MB_OK |
MB_ICONSTOP);
50                   return FALSE;
51               }
52               if ( SQLSetEnvAttr(henv,
SQL_ATTR_CONNECTION_POOLING, (PTR)SQL_CP_ONE_PER_HENV,
SQL_INTEGER) == SQL_ERROR )
53               {
54                   MessageBox(NULL, "Error SQLSetEn-
vAttr() SQL_ATTR_CONNECTION_POOLING", "Init", MB_OK |
MB_ICONSTOP);
55                   return FALSE;
56               }
57           }
58       #endif
59   #else
```

```
60        extern short iMaxConnections;
61
62        dbinit();
63        if ( dbgetmaxprocs() < iMaxConnections )
64        {
65            if ( dbsetmaxprocs(iMaxConnections) == FAIL
)
66            {
67               //set for fail error message when HttpEx-
tensionProc() is called because
68                //at this point we don't have a pECB so
no way to show error message.
69                 iMaxConnections = -1;
70            }
71        }
72        // install error and message handlers
73        dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
74        dberrhandle((DBERRHANDLE_PROC)err_handler);
75    #endif
76
77        InitializeCriticalSection(&ErrorLogCriticalSec-
tion);
78        return TRUE;
79    }
80
81    void
82    SQLCleanup(void)
83    {
84    #ifdef USE_ODBC
85        extern HENV henv;
86        SQLFreeEnv(henv);
87    #else
88        dbexit();
89    #endif
90            DeleteCriticalSection(&ErrorLogCriticalSec-
tion);
91    }
92
93
94
95    /* FUNCTION: int err_handler(DBPROCESS *dbproc, int
severity, int dberr, int oserr, char *dberrstr, char
*oserrstr)
96     *
97     * PURPOSE: This function handles DB-Library errors
98     *
99     * ARGUMENTS:   DBPROCESS       *dbproc
DBPROCESS id pointer
100    *             int            severity       sever-
ity of error
101    *             int            dberr          error
id
102    *              int            oserr          oper-
ating system specific error code
103    *             char           *dberrstr      print-
able error description of dberr
104    *             char           *oserrstr      print-
able error description of oserr
105    *
106    * RETURNS:     int             INT_CONTINUE    con-
tinue if error is SQLETIME else INT_CANCEL action
107    *
108    * COMMENTS:    None
109    *
110    */
111
112   #ifndef USE_ODBC
113      int err_handler(DBPROCESS *dbproc, int severity,
int dberr, int oserr, char *dberrstr, char *oserrstr)
114       {
115           PECBINFO                pEcbInfo;
116           EXTENSION_CONTROL_BLOCK *pECB;
117           FILE                    *fp;
118           SYSTEMTIME              systemTime;
119           char                    szTmp[256];
120           int                     iTermId;
121           int                     iSyncId;
122           pEcbInfo = NULL;
123
124
125           if ((dbproc == NULL) || (DBDEAD(dbproc)))
126           {
127               ErrorMessage(gpECB, -1, ERR_TYPE_DBLIB,
 "DBPROC is invalid.", iTermId, iSyncId);
128               return INT_CANCEL;
129           }
130
131           if ( !(pEcbInfo = (PECBINFO)dbgetuser-
data(dbproc)) )
132           {
133               pECB = gpECB;
134               iTermId = 0;
135               iSyncId = 0;
136           }
137           else
138           {
139               pECB = pEcbInfo->pECB;
140               iTermId = pEcbInfo->iTermId;
141               iSyncId = pEcbInfo->iSyncId;
142           }
143
144           if ( pEcbInfo && pEcbInfo->bFailed )
145               return INT_CANCEL;
146
147           if ( oserr != DBNOERR )
148           {
149               ErrorMessage(pECB, oserr,
ERR_TYPE_DBLIB, oserrstr, iTermId, iSyncId);
150
151               if ( pEcbInfo )
152                   pEcbInfo->bFailed = TRUE;
153
154               GetLocalTime(&systemTime);
155               fp = fopen(szErrorLogPath, "ab");
156
157               EnterCriticalSection(&ErrorLogCritical-
Section);
158
159               sprintf(szTmp, "Error: DBLIB(%d): %s",
oserr, oserrstr);
160
161               fprintf(fp, "%2.2d/%2.2d/%2.2d
%2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
162                  systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
163                systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
164                  szTmp);
165               LeaveCriticalSection(&ErrorLogCritical-
Section);
166
167               fclose(fp);
168           }
169
170           return INT_CANCEL;
171       }
172   #endif
173
174   /* FUNCTION: int msg_handler(DBPROCESS *dbproc,
DBINT msgno, int msgstate, int severity, char *msgtext)
175    *
176    * PURPOSE: This function handles DB-Library SQL
Server error messages
177    *
178    * ARGUMENTS:   DBPROCESS       *dbproc
DBPROCESS id pointer
179    *             DBINT          msgno          message
number
180    *             int            msgstate       message
state
181    *             int            severity       message
severity
182    *              char           *msgtext       print-
able message description
```

```
183     *
184     * RETURNS:     int             INT_CONTINUE    con-
 tinue if error is SQLETIME else INT_CANCEL action
185     *                             INT_CANCEL      cancel
 operation
186     *
187     * COMMENTS:    This function also sets the dead
 lock dbproc variable if necessary.
188     *
189     */
190
191   int msg_handler(DBPROCESS *dbproc, DBINT msgno, int
 msgstate, int severity, char *msgtext)
192   {
193       PECBINFO                pEcbInfo;
194       EXTENSION_CONTROL_BLOCK *pECB;
195       FILE                    *fp;
196       SYSTEMTIME              systemTime;
197       char                    szTmp[256];
198       int                     iTermId;
199       int                     iSyncId;
200
201       if ( !(pEcbInfo = (PECBINFO)dbgetuser-
 data(dbproc)) )
202       {
203           pECB = gpECB;
204           iTermId = 0;
205           iSyncId = 0;
206       }
207       else
208       {
209           pECB = pEcbInfo->pECB;
210           iTermId = pEcbInfo->iTermId;
211           iSyncId = pEcbInfo->iSyncId;
212       }
213
214       if ( (msgno == 5701) || (msgno == 2528) ||
 (msgno == 5703) || (msgno == 6006) )
215           return INT_CONTINUE;
216
217       // deadlock message
218       if (msgno == 1205)
219       {
220           // set the deadlock indicator
221           if ( pEcbInfo )
222               pEcbInfo->bDeadlock = TRUE;
223           else
224               ErrorMessage(pECB, -1, ERR_TYPE_SQL,
 "Error, dbgetuserdata returned NULL.", iTermId, iSyncId);
225           return INT_CONTINUE;
226       }
227       if ( pEcbInfo && pEcbInfo->bFailed )
228           return INT_CANCEL;
229
230       if (msgno == 0)
231           return INT_CONTINUE;
232       else
233       {
234          ErrorMessage(pECB, msgno, ERR_TYPE_SQL, msg-
 text, iTermId, iSyncId);
235
236           if ( pEcbInfo )
237               pEcbInfo->bFailed = TRUE;
238
239           GetLocalTime(&systemTime);
240           fp = fopen(szErrorLogPath, "ab");
241
242           EnterCriticalSection(&ErrorLogCriticalSec-
 tion);
243           sprintf(szTmp, "Error: SQLSVR(%d): %s",
 msgno, msgtext);
244           fprintf(fp, "%2.2d/%2.2d/%2.2d
 %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
245                   systemTime.wYear, systemTime.wMonth,
 systemTime.wDay,
246                   systemTime.wHour, systemTime.wMinute,
 systemTime.wSecond,
247                   szTmp);
248           LeaveCriticalSection(&ErrorLogCriticalSec-
 tion);
249
250           fclose(fp);
251       }
252
253       return INT_CANCEL;
254   }
255
256
257
258
259   /* FUNCTION: BOOL SQLOpenConnec-
 tion(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
 iSyncId, DBPROCESS **dbproc, char *server, char *data-
 base, char *user, char *password, char *app, int *spid,
 long *pack_size)
260     *
261     * PURPOSE: This function opens the sql connection
 for use.
262     *
263     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 passed in structure pointer from inetsrv.
264     *              int             iTermId     terminal
 id of browser
265     *              int             iSyncId     sync id
 of browser
266     *              DBPROCESS       **dbproc    pointer
 to returned DBPROCESS
267     *             char           *server     SQL server
 name
268     *             char           *database   SQL server
 database
269     *              char            *user       user name
270     *             char           *password   user pass-
 word
271     *              char            *app        pointer
 to returned application array
272     *              int             *spid       pointer
 to returned spid
273     *              long            *pack_size  pointer
 to returned default pack size
274     *
275     * RETURNS:     BOOL    FALSE   if successfull
276     *                      TRUE    if an error occurs
277     *
278     * COMMENTS:    None
279     *
280     */
281
282
283   #ifdef USE_ODBC
284       BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK
 *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char
 *server, char *database, char *user, char *password, char
 *app, int *spid)
285       {
286
287           RETCODE     rc;
288           char        buffer[30];
289           PECBINFO    pEcbInfo;
290
291           *dbproc = (DBPROCESS *)malloc(sizeof(DBPRO-
 CESS));
292           if ( !*dbproc )
293               return TRUE;
294
295           //set pECB data into dbproc
296           pEcbInfo = (PECBINFO)mal-
 loc(sizeof(ECBINFO));
297
298           pEcbInfo->bDeadlock = FALSE;
299           pEcbInfo->pECB      = pECB;
300           pEcbInfo->iTermId   = iTermId;
301           pEcbInfo->iSyncId   = iSyncId;
302
```

```
303           dbsetuserdata(*dbproc, pEcbInfo);
304
305          if ( SQLAllocConnect(henv, &(*dbproc)->hdbc)
 == SQL_ERROR )
306           {
307               ODBCError(*dbproc);
308               return TRUE;
309           }
310
311           if ( SQLSetConnectOption((*dbproc)->hdbc,
 SQL_PACKET_SIZE, 4096) == SQL_ERROR )
312           {
313               ODBCError(*dbproc);
314               return TRUE;
315           }
316
317           rc = SQLConnect((*dbproc)->hdbc, server,
 SQL_NTS, user, SQL_NTS, password, SQL_NTS);
318           if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
319           {
320               ODBCError(*dbproc);
321               return TRUE;
322           }
323           rc = SQLAllocStmt((*dbproc)->hdbc,
 &(*dbproc)->hstmt);
324           if (rc == SQL_ERROR)
325           {
326               ODBCError(*dbproc);
327               return TRUE;
328           }
329
330          strcpy(buffer, "use tpcc set nocount on set
 XACT_ABORT ON");
331          rc = SQLExecDirect((*dbproc)->hstmt, buffer,
 SQL_NTS);
332           if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
333           {
334               ODBCError(*dbproc);
335               return TRUE;
336           }
337           SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);
338
339           sprintf(buffer,"select @@spid");
340
341          rc = SQLExecDirect((*dbproc)->hstmt, buffer,
 SQL_NTS);
342           if (rc != SQL_SUCCESS && rc !=
 SQL_SUCCESS_WITH_INFO)
343           {
344               ODBCError(*dbproc);
345               return TRUE;
346           }
347
348           if ( SQLBindCol((*dbproc)->hstmt, 1,
 SQL_C_SSHORT, &(*dbproc)->spid, 0, NULL) == SQL_ERROR )
349           {
350               ODBCError(*dbproc);
351               return TRUE;
352           }
353
354          if ( SQLFetch((*dbproc)->hstmt) == SQL_ERROR
 )
355           {
356               ODBCError(*dbproc);
357               return TRUE;
358           }
359
360           SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);
361
362           if ( bConnectionPooling )
363               SQLDisconnect((*dbproc)->hdbc);
364
365           return FALSE;
366       }
367   #else
368
369       BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK
 *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char
 *server, char *database, char *user, char *password, char
 *app, int *spid)
370       {
371           LOGINREC    *login;
372           PECBINFO    pEcbInfo;
373
374           //set local msg proc for login record
375           //attach pECB record
376
377          //this is necessary as dblib provides no way
 to pass user data in a login structure. So until
378          //there is an allocated dbproc we need to use
 a static which means that the login attempt must
379           //be serialized.
380
381           gpECB = pECB;
382
383           login = dblogin();
384
385           if ( !*user )
386               DBSETLUSER(login, "sa");
387           else
388               DBSETLUSER(login, user);
389
390           DBSETLPWD(login,  password);
391           DBSETLHOST(login, app);
392
393           DBSETLPACKET(login, (unsigned short)DEF-
 CLPACKSIZE);
394           if ((*dbproc = dbopen(login, server )) ==
 NULL)
395               return TRUE;
396
397           //set pECB data into dbproc
398           pEcbInfo = (PECBINFO)mal-
 loc(sizeof(ECBINFO));
399           pEcbInfo->bDeadlock = FALSE;
400           pEcbInfo->pECB = pECB;
401           pEcbInfo->iTermId = iTermId;
402           pEcbInfo->iSyncId = iSyncId;
403           dbsetuserdata(*dbproc, pEcbInfo);
404
405           // Use the the right database
406           dbuse(*dbproc, database);
407
408           dbcmd(*dbproc, "select @@spid");
409           dbsqlexec(*dbproc);
410
411           while (dbresults(*dbproc) !=
 NO_MORE_RESULTS)
412           {
413              dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0,
  (BYTE *) spid);
414               while (dbnextrow(*dbproc) !=
 NO_MORE_ROWS)
415                   ;
416           }
417
418           dbcmd(*dbproc, "set nocount on");
419           dbsqlexec(*dbproc);
420           while (dbresults(*dbproc) !=
 NO_MORE_RESULTS)
421           {
422               while (dbnextrow(*dbproc) !=
 NO_MORE_ROWS)
423                   ;
424           }
425
426           //rollback transaction on abort
427           dbcmd(*dbproc, "set XACT_ABORT ON");
428           dbsqlexec(*dbproc);
429           while (dbresults(*dbproc) !=
 NO_MORE_RESULTS)
430           {
```

```
431                 while (dbnextrow(*dbproc) !=
 NO_MORE_ROWS)
432                     ;
433             }
434
435             return FALSE;
436         }
437
438     #endif
439
440     /* FUNCTION: BOOL SQLCloseConnec-
 tion(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS *dbproc)
441     *
442     * PURPOSE: This function closes the sql connection.
443     *
444     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 passed in structure pointer from inetsrv.
445     *              DBPROCESS       *dbproc pointer to
 DBPROCESS
446     *
447     * RETURNS:     BOOL    FALSE   if successfull
448     *                      TRUE    if an error occurs
449     *
450     * COMMENTS:    None
451     *
452     */
453
454     #ifdef USE_ODBC
455         BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK
 *pECB, DBPROCESS *dbproc)
456         {
457             if ( dbproc )
458             {
459                 SQLFreeStmt(dbproc->hstmt, SQL_DROP);
460                 SQLDisconnect(dbproc->hdbc);
461                 SQLFreeConnect(dbproc->hdbc);
462                 free(dbproc);
463                 dbproc = NULL;
464             }
465             return FALSE;
466         }
467     #else
468         BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK
 *pECB, DBPROCESS *dbproc)
469         {
470             if (dbclose(dbproc) == FAIL)
471                 return TRUE;
472             return FALSE;
473         }
474     #endif
475
476     /* FUNCTION: SQLStock-
 Level(EXTENSION_CONTROL_BLOCK  *pECB, int iTermId, int
 iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStock-
 Level, short deadlock_retry)
477     *
478     * PURPOSE: This function handles the stock level
 transaction.
479     *
480     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK
 *pECB           passed in structure pointer from inetsrv.
481     *              int                     iTer-
 mId         terminal id of browser
482     *              int                     iSyn-
 cId         sync id of browser
483     *              DBPROCESS
 *dbproc         connection db process id
484     *            STOCK_LEVEL_DATA      *pStockLevel
 stock level input / output data structure
485     *             short                deadlock_retry
 retry count if deadlocked
486     *
487     * RETURNS:     BOOL    FALSE       if successfull
488     *                      TRUE        if deadlocked
489     *
490     * COMMENTS:    None
491     *
492     */
493
494     #ifdef USE_ODBC
495         int SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB,
 int iTermId, int iSyncId, DBPROCESS *dbproc,
 STOCK_LEVEL_DATA *pStockLevel, short deadlock_retry)
496         {
497             int         tryit;
498             PECBINFO    pEcbInfo;
499
500
501             //update pECB and bFailed flag
502             if ( (pEcbInfo = (PECBINFO)dbgetuser-
 data(dbproc)) )
503             {
504                 pEcbInfo->pECB = pECB;
505                 pEcbInfo->bFailed = FALSE;
506                 pEcbInfo->iTermId = iTermId;
507                 pEcbInfo->iSyncId = iSyncId;
508             }
509
510     #ifdef USE_ODBC
511             if ( ReopenConnection(dbproc) )
512                 return -3;
513     #endif
514
515             pStockLevel->num_deadlocks = 0;
516
517             for (tryit=0; tryit<deadlock_retry; tryit++)
518             {
519                 BindParameter(dbproc, 1,SQL_C_SSHORT,
 SQL_SMALLINT, 0, 0, &pStockLevel->w_id, 0);
520                 BindParameter(dbproc, 2,SQL_C_STINYINT,
 SQL_TINYINT, 0, 0, &pStockLevel->d_id, 0);
521                 BindParameter(dbproc, 3,SQL_C_SSHORT,
 SQL_SMALLINT, 0, 0, &pStockLevel->thresh_hold, 0);
522
523                 if ( !ExecuteStatement(dbproc, "{call
 tpcc_stocklevel(?,?,?)}") )
524                 {
525                     if ( !SQLDetectDeadlock(dbproc) )
526                     {
527                         if ( BindColumn(dbproc, 1,
 SQL_C_SSHORT, &pStockLevel->low_stock, 0) )
528                             return TRUE;
529
530                         if ( GetResults(dbproc) )
531                             return TRUE;
532                     }
533                 }
534
535                 SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
536
537                 if ( SQLDetectDeadlock(dbproc) )
538                 {
539                     pStockLevel->num_deadlocks++;
540                     Sleep(10 * tryit);
541                 }
542                 else
543                 {
544                   strcpy(pStockLevel->execution_status,
 "Transaction commited.");
545                     return FALSE;
546                 }
547             }
548
549           // If we reached here, it means we quit after
 MAX_RETRY deadlocks
550             strcpy(pStockLevel->execution_status, "Hit
 deadlock max.");
551
552             return TRUE;
553         }
554     #else
555         BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK
 *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc,
 STOCK_LEVEL_DATA *pStockLevel, short deadlock_retry)
```

```
556       {
557           int         tryit;
558           RETCODE     rc;
559           char        printbuf[25];
560           BYTE        *pData;
561           PECBINFO    pEcbInfo;
562
563           //update pECB and bFailed flag
564           if ( (pEcbInfo = (PECBINFO)dbgetuser-
 data(dbproc)) )
565           {
566               pEcbInfo->pECB = pECB;
567               pEcbInfo->bFailed = FALSE;
568               pEcbInfo->iTermId = iTermId;
569               pEcbInfo->iSyncId = iSyncId;
570           }
571
572           pStockLevel->num_deadlocks = 0;
573
574           for (tryit=0; tryit < deadlock_retry;
 tryit++)
575           {
576              if (dbrpcinit(dbproc, "tpcc_stocklevel",
 0) == SUCCEED)
577               {
578                  dbrpcparam(dbproc, NULL, 0, SQLINT2,
 -1, -1, (BYTE *) &pStockLevel->w_id);
579                  dbrpcparam(dbproc, NULL, 0, SQLINT1,
 -1, -1, (BYTE *) &pStockLevel->d_id);
580                  dbrpcparam(dbproc, NULL, 0, SQLINT2,
 -1, -1, (BYTE *) &pStockLevel->thresh_hold);
581
582                  if (dbrpcexec(dbproc) == SUCCEED)
583                  {
584                     while (((rc = dbresults(dbproc))
 != NO_MORE_RESULTS) && (rc != FAIL))
585                      {
586                          if (DBROWS(dbproc))
587                          {
588                              while (((rc = dbnex-
 trow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
589                              {
590
 if(pData=dbdata(dbproc, 1))
591                                  pStockLevel-
 >low_stock = *((long *) pData);
592                              }
593                          }
594                      }
595                  }
596               }
597               if (SQLDetectDeadlock(dbproc))
598               {
599                   pStockLevel->num_deadlocks++;
600                   sprintf(printbuf,"deadlock: retry:
 %d",pStockLevel->num_deadlocks);
601                   Sleep(10 * tryit);
602               }
603               else
604               {
605                strcpy(pStockLevel->execution_status,
 "Transaction commited.");
606                   return FALSE;
607               }
608           }
609
610         // If we reached here, it means we quit after
 MAX_RETRY deadlocks
611           strcpy(pStockLevel->execution_status, "Hit
 deadlock max. ");
612           return TRUE;
613       }
614   #endif
615
616   /* FUNCTION: int SQLNe-
 wOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
 iSyncId, int iTermId, int iSyncId, DBPROCESS *dbproc,
 NEW_ORDER_DATA  *pNewOrder, short deadlock_retry)
617    *
618    * PURPOSE: This function handles the new order
 transaction.
619    *
620    * ARGUMENTS:   EXTENSION_CONTROL_BLOCK
 *pECB          passed in structure pointer from inetsrv.
621    *             int                       iTer-
 mId         terminal id of browser
622    *             int                       iSyn-
 cId         sync id of browser
623    *             DBPROCESS
 *dbproc         connection db process id
624    *           NEW_ORDER_DATA         *pNewOrder
 pointer to new order structure for input/output data
625    *            short                deadlock_retry
 retry count if deadlocked
626    *
627    * RETURNS:     int TRUE    transaction committed
628    *                  FALSE   item number not valid
629    *                -1      deadlock max retry reached
630    *
631    *
632    * COMMENTS:    None
633    *
634    */
635
636   #ifdef USE_ODBC
637       int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB,
 int iTermId, int iSyncId, DBPROCESS *dbproc,
 NEW_ORDER_DATA  *pNewOrder, short deadlock_retry)
638       {
639           int         i;
640           int         j;
641           int         tryit;
642           DBINT       commit_flag;
643           char        buffer[255];
644           PECBINFO    pEcbInfo;
645
646           if ( (pEcbInfo = (PECBINFO)dbgetuser-
 data(dbproc)) )
647           {
648               pEcbInfo->pECB = pECB;
649               pEcbInfo->bFailed = FALSE;
650               pEcbInfo->iTermId = iTermId;
651               pEcbInfo->iSyncId = iSyncId;
652           }
653
654           if ( ReopenConnection(dbproc) )
655               return -3;
656
657           pNewOrder->num_deadlocks = 0;
658
659          for (tryit=0; tryit<deadlock_retry; tryit++)
660           {
661               strcpy(buffer, "{call
 tpcc_neworder(?,?,?,?,?,");
662               for (i=1; i<pNewOrder->o_ol_cnt; i++)
663                   strcat(buffer, "?,?,?,");
664               strcat(buffer, "?,?,?)}");
665
666              BindParameter(dbproc, 1, SQL_C_SSHORT,
 SQL_SMALLINT,   0, 0, &pNewOrder->w_id, 0);
667              BindParameter(dbproc, 2, SQL_C_STINYINT,
 SQL_TINYINT,    0, 0, &pNewOrder->d_id, 0);
668              BindParameter(dbproc, 3, SQL_C_SLONG,
 SQL_INTEGER,    0, 0, &pNewOrder->c_id, 0);
669              BindParameter(dbproc, 4, SQL_C_STINYINT,
 SQL_TINYINT,    0, 0, &pNewOrder->o_ol_cnt, 0);
670
671               pNewOrder->o_all_local = 1;
672               for (j=0; j<pNewOrder->o_ol_cnt; j++)
673               {
674                   if ( pNewOrder->o_all_local && pNe-
 wOrder->Ol[j].ol_supply_w_id != pNewOrder->w_id )
675                       pNewOrder->o_all_local = 0;
676               }
```

```
677                 BindParameter(dbproc, 5, SQL_C_STINYINT,
 SQL_TINYINT,    0, 0, &pNewOrder->o_all_local, 0);
678
679                 for (j=0, i=0; i<(pNewOrder->o_ol_cnt *
 3); i=i+3, j++)
680                 {
681                     BindParameter(dbproc, (UWORD)(i+6),
 SQL_C_SLONG,  SQL_INTEGER,  0, 0, &pNewOrder-
 >Ol[j].ol_i_id, 0);
682                     BindParameter(dbproc, (UWORD)(i+7),
 SQL_C_SSHORT, SQL_SMALLINT, 0, 0, &pNewOrder-
 >Ol[j].ol_supply_w_id, 0);
683                     BindParameter(dbproc, (UWORD)(i+8),
 SQL_C_SSHORT, SQL_SMALLINT, 0, 0, &pNewOrder-
 >Ol[j].ol_quantity, 0);
684                 }
685
686                 if ( ExecuteStatement(dbproc, buffer) )
687                     if ( !SQLDetectDeadlock(dbproc) )
688                         return -2;
689
690
691                 pNewOrder->total_amount=0;
692
693                 for (i = 0; i<pNewOrder->o_ol_cnt; i++)
694                 {
695                  if ( BindColumn(dbproc,1, SQL_C_CHAR,
 &pNewOrder->Ol[i].ol_i_name, sizeof(pNewOrder-
 >Ol[i].ol_i_name)) )
696                         return -2;
697                     if ( BindColumn(dbproc,2,
 SQL_C_SSHORT, &pNewOrder->Ol[i].ol_stock, 0) )
698                         return -2;
699                  if ( BindColumn(dbproc,3, SQL_C_CHAR,
 &pNewOrder->Ol[i].ol_brand_generic, sizeof(pNewOrder-
 >Ol[i].ol_brand_generic)) )
700                         return -2;
701                     if ( BindColumn(dbproc,4,
 SQL_C_DOUBLE, &pNewOrder->Ol[i].ol_i_price, 0) )
702                         return -2;
703                     if ( BindColumn(dbproc,5,
 SQL_C_DOUBLE, &pNewOrder->Ol[i].ol_amount, 0) )
704                         return -2;
705
706                     if ( GetResults(dbproc) )
707                         return -2;
708
709                    pNewOrder->total_amount = pNewOrder-
 >total_amount + pNewOrder->Ol[i].ol_amount;
710                     if ( !pEcbInfo->bDeadlock )
711                     {
712                         if ( MoreResults(dbproc) )
713                             return -2;
714                     }
715                     if ( pEcbInfo->bDeadlock )
716                         break;
717                 }
718
719                 if ( !SQLDetectDeadlock(dbproc) )
720                 {
721                     if ( BindColumn(dbproc, 1,
 SQL_C_DOUBLE, &pNewOrder->w_tax, 0) )
722                         return -2;
723                     if ( BindColumn(dbproc, 2,
 SQL_C_DOUBLE,    &pNewOrder->d_tax, 0) )
724                         return -2;
725                     if ( BindColumn(dbproc, 3,
 SQL_C_SLONG,     &pNewOrder->o_id, 0) )
726                         return -2;
727                     if ( BindColumn(dbproc, 4,
 SQL_C_CHAR,      &pNewOrder->c_last, sizeof(pNewOrder-
 >c_last)) )
728                         return -2;
729                     if ( BindColumn(dbproc, 5,
 SQL_C_DOUBLE,    &pNewOrder->c_discount, 0) )
730                         return -2;
731                     if ( BindColumn(dbproc, 6,
 SQL_C_CHAR,      &pNewOrder->c_credit, sizeof(pNewOrder-
 >c_credit)) )
732                         return -2;
733                     if ( BindColumn(dbproc, 7,
 SQL_C_TIMESTAMP, &pNewOrder->o_entry_d, 0) )
734                         return -2;
735                     if ( BindColumn(dbproc, 8,
 SQL_C_SLONG,     &commit_flag, 0) )
736                         return -2;
737
738                     if ( GetResults(dbproc) )
739                         return -2;
740
741                     SQLFreeStmt(dbproc->hstmt,
 SQL_CLOSE);
742
743                     if ( commit_flag == 1 )
744                     {
745                         pNewOrder->total_amount = pNe-
 wOrder->total_amount * ((1 + pNewOrder->w_tax + pNe-
 wOrder->d_tax) * (1 - pNewOrder->c_discount));
746                         strcpy(pNewOrder-
 >execution_status,"Transaction commited.");
747                         return TRUE;
748                     }
749                     else
750                     {
751                         strcpy(pNewOrder-
 >execution_status,"Item number is not valid.");
752                         return FALSE;
753                     }
754                 }
755                 else
756                 {
757                     SQLFreeStmt(dbproc->hstmt,
 SQL_CLOSE);
758                     pNewOrder->num_deadlocks++;
759                     Sleep(DEADLOCKWAIT*tryit);
760                 }
761             }
762           // If we reached here, it means we quit after
 MAX_RETRY deadlocks
763             strcpy(pNewOrder->execution_status,"Hit
 deadlock max.  ");
764             return -1;
765         }
766    #else
767         int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB,
 int iTermId, int iSyncId, DBPROCESS *dbproc,
 NEW_ORDER_DATA  *pNewOrder, short deadlock_retry)
768         {
769             RETCODE         rc;
770             int             i;
771             DBINT           commit_flag;
772             int             tryit;
773             char            printbuf[25];
774             char            tmpbuf[30];
775             DBDATETIME      datetime;
776             BYTE            *pData;
777             PECBINFO        pEcbInfo;
778    #ifdef EXTRA_DEBUG
779             char ExtraDebugBuf[8192]={0};
780
781             sprintf(ExtraDebugBuf, "new_order begins
 w_id = %d, d_id=%d, c_id=%d o_ol_cnt=%d\n",
782                         pNewOrder->w_id,
783                         pNewOrder->d_id,
784                         pNewOrder->c_id,
785                         pNewOrder->o_ol_cnt);
786    #endif
787             if ( (pEcbInfo = (PECBINFO)dbgetuser-
 data(dbproc)) )
788             {
789                 pEcbInfo->pECB = pECB;
790                 pEcbInfo->bFailed = FALSE;
791                 pEcbInfo->iTermId = iTermId;
792                 pEcbInfo->iSyncId = iSyncId;
```

```
793           }
794
795           pNewOrder->num_deadlocks = 0;
796
797           strcpy(tmpbuf, "tpcc_neworder");
798
799           for (tryit=0; tryit < deadlock_retry;
 tryit++)
800           {
801   #ifdef EXTRA_DEBUG
802               if (tryit)
803                  sprintf(ExtraDebugBuf+strlen(ExtraDe-
 bugBuf), "\tretry %d\n");
804   #endif
805               if (dbrpcinit(dbproc, tmpbuf, 0) == SUC-
 CEED)
806               {
807                  dbrpcparam(dbproc, NULL, 0, SQLINT2,
 -1, -1, (BYTE *) &pNewOrder->w_id);
808                  dbrpcparam(dbproc, NULL, 0, SQLINT1,
 -1, -1, (BYTE *) &pNewOrder->d_id);
809                  dbrpcparam(dbproc, NULL, 0, SQLINT4,
 -1, -1, (BYTE *) &pNewOrder->c_id);
810                  dbrpcparam(dbproc, NULL, 0, SQLINT1,
 -1, -1, (BYTE *) &pNewOrder->o_ol_cnt);
811
812                   pNewOrder->o_all_local = 1;
813                  for (i = 0; i < pNewOrder->o_ol_cnt;
 i++)
814                   {
815                       if ( pNewOrder->o_all_local &&
 pNewOrder->Ol[i].ol_supply_w_id != pNewOrder->w_id )
816                           pNewOrder->o_all_local = 0;
817                   }
818                  dbrpcparam(dbproc, NULL, 0, SQLINT1,
 -1, -1, (BYTE *) &pNewOrder->o_all_local);
819
820                  for (i = 0; i < pNewOrder->o_ol_cnt;
 i++)
821                   {
822   #ifdef EXTRA_DEBUG
823                       sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "\ti=%d ol_i_id=%d,
 ol_supply_w_id=%d, ol_quantity=%d\n", i, pNewOrder-
 >Ol[i].ol_i_id, pNewOrder->Ol[i].ol_supply_w_id,  pNe-
 wOrder->Ol[i].ol_quantity);
824   #endif
825                       dbrpcparam(dbproc, NULL, 0,
 SQLINT4, -1, -1, (BYTE *) &pNewOrder->Ol[i].ol_i_id);
826                       dbrpcparam(dbproc, NULL, 0,
 SQLINT2, -1, -1, (BYTE *) &pNewOrder-
 >Ol[i].ol_supply_w_id);
827                       dbrpcparam(dbproc, NULL, 0,
 SQLINT2, -1, -1, (BYTE *) &pNewOrder->Ol[i].ol_quantity);
828                   }
829
830                   if (dbrpcexec(dbproc) == SUCCEED)
831                   {
832   #ifdef EXTRA_DEBUG
833                       sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "\tdbrpcexec succeeded, ret-
 status=%d\n", dbretstatus(dbproc));
834   #endif
835                       pNewOrder->total_amount=0;
836
837                       // Get resutls from order line
838                       for (i = 0; i<pNewOrder-
 >o_ol_cnt; i++)
839                       {
840   #ifdef EXTRA_DEBUG
841                           sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "\tresults line %d ", i);
842   #endif
843                         if (((rc = dbresults(dbproc))
 != NO_MORE_RESULTS) && (rc != FAIL))
844                           {
845   #ifdef EXTRA_DEBUG
846                               sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "dbnumcols=%d\n", dbnum-
 cols(dbproc));
847   #endif
848                               if (DBROWS(dbproc) &&
 (dbnumcols(dbproc) == 5))
849                               {
850                                   while (dbnex-
 trow(dbproc) != NO_MORE_ROWS)
851                                   {
852
 if(pData=dbdata(dbproc, 1))
853                                       UtilStrCpy(pNe-
 wOrder->Ol[i].ol_i_name, pData, dbdatlen(dbproc, 1));
854
 if(pData=dbdata(dbproc, 2))
855                                       pNewOrder-
 >Ol[i].ol_stock = (*(DBSMALLINT *) pData);
856
 if(pData=dbdata(dbproc, 3))
857                                       UtilStrCpy(pNe-
 wOrder->Ol[i].ol_brand_generic, pData, dbdatlen(dbproc,
 3));
858
 if(pData=dbdata(dbproc, 4))
859                                       pNewOrder-
 >Ol[i].ol_i_price = (*(DBFLT8 *) pData);
860
 if(pData=dbdata(dbproc, 5))
861                                       pNewOrder-
 >Ol[i].ol_amount = (*(DBFLT8 *) pData);
862                                     pNewOrder-
 >total_amount = pNewOrder->total_amount + pNewOrder-
 >Ol[i].ol_amount;
863                                   }
864                               }
865                           }
866   #ifdef EXTRA_DEBUG
867                           else
868                             if (rc != NO_MORE_RESULTS)
869                                 sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "for loop dbresults returned
 rc=%d\n", rc);
870   #endif
871                       }
872   #ifdef EXTRA_DEBUG
873                       sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "\tstarting while");
874   #endif
875                       while (((rc = dbresults(dbproc))
 != NO_MORE_RESULTS) && (rc != FAIL))
876                       {
877   #ifdef EXTRA_DEBUG
878                           sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf)," dbnumcols=%d", dbnum-
 cols(dbproc));
879   #endif
880                         if (DBROWS(dbproc) && (dbnum-
 cols(dbproc) == 8))
881                         {
882                             while (((rc = dbnex-
 trow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
883                             {
884
 if(pData=dbdata(dbproc, 1))
885                                 pNewOrder->w_tax
 = (*(DBFLT8 *) pData);
886
 if(pData=dbdata(dbproc, 2))
887                                 pNewOrder->d_tax
 = (*(DBFLT8 *) pData);
888
 if(pData=dbdata(dbproc, 3))
889                                  pNewOrder->o_id
 = (*(DBINT *) pData);
890
 if(pData=dbdata(dbproc, 4))
```

```
891                                     UtilStrCpy(pNe-
 wOrder->c_last, pData, dbdatlen(dbproc, 4));
892
 if(pData=dbdata(dbproc, 5))
893                                     pNewOrder-
 >c_discount = (*(DBFLT8 *) pData);
894
 if(pData=dbdata(dbproc, 6))
895                                     UtilStrCpy(pNe-
 wOrder->c_credit, pData, dbdatlen(dbproc, 6));
896
 if(pData=dbdata(dbproc, 7))
897                                    {
898                                     datetime = *((DBDA-
 TETIME *) pData);
899                                    dbdatecrack(dbproc,
 &pNewOrder->o_entry_d, &datetime);
900                                    }
901
 if(pData=dbdata(dbproc, 8))commit_flag = (*(DBTINYINT *)
 pData);
902                                 }
903                             }
904                         }
905   #ifdef EXTRA_DEBUG
906                         sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), " second while ends, rc=%d
 commit_flag = %d\n", rc, commit_flag);
907   #endif
908                     }
909   #ifdef EXTRA_DEBUG
910                     else
911                         sprintf(ExtraDebug-
 Buf+strlen(ExtraDebugBuf), "\tdbrpcexe != SUCCESS\n");
912   #endif
913                 }
914   #ifdef EXTRA_DEBUG
915                 else
916                    sprintf(ExtraDebugBuf+strlen(ExtraDe-
 bugBuf), "dbrpcinit failed\n");
917   #endif
918                 if (SQLDetectDeadlock(dbproc))
919                 {
920                     pNewOrder->num_deadlocks++;
921                     sprintf(printbuf,"deadlock: retry:
 %d",pNewOrder->num_deadlocks);
922                     Sleep(DEADLOCKWAIT*tryit);
923                 }
924                 else
925                 {
926   #ifdef EXTRA_DEBUG
927                     EnterCriticalSection(&ErrorLogCrit-
 icalSection);
928                     fputs(ExtraDebugBuf, stderr);
929                     LeaveCriticalSection(&ErrorLogCrit-
 icalSection);
930   #endif
931                     if (commit_flag == 1)
932                     {
933                         pNewOrder->total_amount = pNe-
 wOrder->total_amount * ((1 + pNewOrder->w_tax + pNe-
 wOrder->d_tax) * (1 - pNewOrder->c_discount));
934                         strcpy(pNewOrder-
 >execution_status,"Transaction commited.");
935
936                         return TRUE;
937                     }
938                     else
939                     {
940                         strcpy(pNewOrder-
 >execution_status,"Item number is not valid.");
941                         return FALSE;
942                     }
943                 }
944           }
945
946          // If we reached here, it means we quit after
 MAX_RETRY deadlocks
947           strcpy(pNewOrder->execution_status,"Hit
 deadlock max.  ");
948   #ifdef EXTRA_DEBUG
949                   EnterCriticalSection(&ErrorLogCrit-
 icalSection);
950                   fputs(ExtraDebugBuf, stderr);
951                   LeaveCriticalSection(&ErrorLogCrit-
 icalSection);
952   #endif
953          return -1;  //  "deadlock max retry reached!"
954       }
955   #endif
956
957   /* FUNCTION: int SQLPay-
 ment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
 iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
 deadlock_retry)
958    *
959    * PURPOSE: This function handles the payment trans-
 action.
960    *
961    * ARGUMENTS:   EXTENSION_CONTROL_BLOCK
 *pECB           passed in structure pointer from inetsrv.
962    *              int                       iTer-
 mId         terminal id of browser
963    *              int                       iSyn-
 cId         sync id of browser
964    *              DBPROCESS
 *dbproc         connection db process id
965    *            PAYMENT_DATA           *pPayment
 pointer to payment input/output data structure
966    *              short                 deadlock_retry
 deadlock retry count
967    *
968    * RETURNS:     int TRUE        success
969    *                -1         max deadlocked reached
970    *
971    * COMMENTS:    None
972    *
973    */
974
975   #ifdef USE_ODBC
976       int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB,
 int iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA
 *pPayment, short deadlock_retry)
977       {
978           int         tryit;
979           char        printbuf[25];
980           char        buffer[255];
981           BOOL        deadlock_detected;
982           PECBINFO    pEcbInfo;
983
984           if ( (pEcbInfo = (PECBINFO)dbgetuser-
 data(dbproc)) )
985           {
986               pEcbInfo->pECB = pECB;
987               pEcbInfo->bFailed = FALSE;
988               pEcbInfo->iTermId = iTermId;
989               pEcbInfo->iSyncId = iSyncId;
990           }
991
992           if ( ReopenConnection(dbproc) )
993               return -3;
994
995           pPayment->num_deadlocks = 0;
996
997           for (tryit=0; tryit<deadlock_retry; tryit++)
998           {
999
1000              deadlock_detected = FALSE;
1001
1002              strcpy(buffer,"{call
  tpcc_payment(?,?,?,?,?,?");
1003              if (pPayment->c_id == 0)
1004                  strcat(buffer,",?");
1005              strcat(buffer,")}");
```

```
1006
1007                BindParameter(dbproc, 1, SQL_C_SSHORT,
  SQL_SMALLINT, 0,       0, &pPayment->w_id,     0);
1008                BindParameter(dbproc, 2, SQL_C_SSHORT,
  SQL_SMALLINT, 0,       0, &pPayment->c_w_id,   0);
1009                BindParameter(dbproc, 3, SQL_C_DOUBLE,
  SQL_NUMERIC,  6,       2, &pPayment->h_amount, 0);
1010               BindParameter(dbproc, 4, SQL_C_STINYINT,
  SQL_TINYINT,  0,       0, &pPayment->d_id,     0);
1011                BindParameter(dbproc, 5, SQL_C_STINYINT,
  SQL_TINYINT,  0,       0, &pPayment->c_d_id,   0);
1012                BindParameter(dbproc, 6, SQL_C_SLONG,
  SQL_INTEGER, (UINT)SQL_NTS, 0, &pPayment->c_id,     0);
1013
1014                if (pPayment->c_id == 0)
1015                   BindParameter(dbproc, 7, SQL_C_CHAR,
  SQL_CHAR, (UINT)SQL_NTS, 0, &pPayment->c_last,
  sizeof(pPayment->c_last));
1016
1017                if ( ExecuteStatement(dbproc, buffer) )
1018                    if ( !pEcbInfo->bDeadlock )
1019                        return -2;
1020
1021                if ( !pEcbInfo->bDeadlock )
1022                {
1023                    if ( BindColumn(dbproc, 1,
  SQL_C_SLONG, &pPayment->c_id, 0) )
1024                        return -2;
1025                    if ( BindColumn(dbproc, 2,
  SQL_C_CHAR, &pPayment->c_last, sizeof(pPayment-
  >c_last)) )
1026                        return -2;
1027                    if ( BindColumn(dbproc, 3,
  SQL_C_TIMESTAMP, &pPayment->h_date, 0) )
1028                        return -2;
1029                    if ( BindColumn(dbproc, 4,
  SQL_C_CHAR, &pPayment->w_street_1, sizeof(pPayment-
  >w_street_1)) )
1030                        return -2;
1031                    if ( BindColumn(dbproc, 5,
  SQL_C_CHAR, &pPayment->w_street_2, sizeof(pPayment-
  >w_street_2)) )
1032                        return -2;
1033                    if ( BindColumn(dbproc, 6,
  SQL_C_CHAR, &pPayment->w_city, sizeof(pPayment-
  >w_city)) )
1034                        return -2;
1035                    if ( BindColumn(dbproc, 7,
  SQL_C_CHAR, &pPayment->w_state, sizeof(pPayment-
  >w_state)) )
1036                        return -2;
1037                    if ( BindColumn(dbproc, 8,
  SQL_C_CHAR, &pPayment->w_zip, sizeof(pPayment->w_zip)) )
1038                        return -2;
1039                    if ( BindColumn(dbproc, 9,
  SQL_C_CHAR, &pPayment->d_street_1, sizeof(pPayment-
  >d_street_1)) )
1040                        return -2;
1041                    if ( BindColumn(dbproc, 10,
  SQL_C_CHAR, &pPayment->d_street_2, sizeof(pPayment-
  >d_street_2)) )
1042                        return -2;
1043                    if ( BindColumn(dbproc, 11,
  SQL_C_CHAR, &pPayment->d_city, sizeof(pPayment-
  >d_city)) )
1044                        return -2;
1045                    if ( BindColumn(dbproc, 12,
  SQL_C_CHAR, &pPayment->d_state, sizeof(pPayment-
  >d_state)) )
1046                        return -2;
1047                    if ( BindColumn(dbproc, 13,
  SQL_C_CHAR, &pPayment->d_zip, sizeof(pPayment->d_zip)) )
1048                        return -2;
1049                    if ( BindColumn(dbproc, 14,
  SQL_C_CHAR, &pPayment->c_first, sizeof(pPayment-
  >c_first)) )
1050                        return -2;
1051                    if ( BindColumn(dbproc, 15,
  SQL_C_CHAR, &pPayment->c_middle, sizeof(pPayment-
  >c_middle)) )
1052                        return -2;
1053                    if ( BindColumn(dbproc, 16,
  SQL_C_CHAR, &pPayment->c_street_1, sizeof(pPayment-
  >c_street_1)) )
1054                        return -2;
1055                    if ( BindColumn(dbproc, 17,
  SQL_C_CHAR, &pPayment->c_street_2, sizeof(pPayment-
  >c_street_2)) )
1056                        return -2;
1057                    if ( BindColumn(dbproc, 18,
  SQL_C_CHAR, &pPayment->c_city, sizeof(pPayment-
  >c_city)) )
1058                        return -2;
1059                    if ( BindColumn(dbproc, 19,
  SQL_C_CHAR, &pPayment->c_state, sizeof(pPayment-
  >c_state)) )
1060                        return -2;
1061                    if ( BindColumn(dbproc, 20,
  SQL_C_CHAR, &pPayment->c_zip, sizeof(pPayment->c_zip)) )
1062                        return -2;
1063                    if ( BindColumn(dbproc, 21,
  SQL_C_CHAR, &pPayment->c_phone, sizeof(pPayment-
  >c_phone)) )
1064                        return -2;
1065                    if ( BindColumn(dbproc, 22,
  SQL_C_TIMESTAMP, &pPayment->c_since, 0) )
1066                        return -2;
1067                    if ( BindColumn(dbproc, 23,
  SQL_C_CHAR, &pPayment->c_credit, sizeof(pPayment-
  >c_credit)) )
1068                        return -2;
1069                    if ( BindColumn(dbproc, 24,
  SQL_C_DOUBLE, &pPayment->c_credit_lim, 0) )
1070                        return -2;
1071                    if ( BindColumn(dbproc, 25,
  SQL_C_DOUBLE, &pPayment->c_discount, 0) )
1072                        return -2;
1073                    if ( BindColumn(dbproc, 26,
  SQL_C_DOUBLE, &pPayment->c_balance, 0) )
1074                        return -2;
1075                    if ( BindColumn(dbproc, 27,
  SQL_C_CHAR, &pPayment->c_data, sizeof(pPayment-
  >c_data)) )
1076                        return -2;
1077
1078                    if ( GetResults(dbproc) )
1079                        return -2;
1080                }
1081
1082                SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
1083
1084                if ( SQLDetectDeadlock(dbproc) )
1085                {
1086                    pPayment->num_deadlocks++;
1087                    sprintf(printbuf,"deadlock: retry:
  %d",pPayment->num_deadlocks);
1088                    Sleep(DEADLOCKWAIT*tryit);
1089                }
1090                else
1091                {
1092                    if ( pPayment->c_id == 0 )
1093                    {
1094                        strcpy(pPayment-
  >execution_status,"Invalid Customer id,name.");
1095                        return 0;
1096                    }
1097                    else
1098                        strcpy(pPayment-
  >execution_status,"Transaction commited.");
1099
1100                    return TRUE;
1101                }
1102            }
1103
```

```
1104          // If we reached here, it means we quit after
  MAX_RETRY deadlocks
1105          strcpy(pPayment->execution_status,"Hit dead-
  lock max.  ");
1106          return -1;
1107       }
1108   #else
1109       int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB,
  int iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA
  *pPayment, short deadlock_retry)
1110       {
1111           RETCODE     rc;
1112           int         tryit;
1113           char        printbuf[26];
1114           DBDATETIME  datetime;
1115           BYTE        *pData;
1116           PECBINFO    pEcbInfo;
1117   #ifdef EXTRA_DEBUG
1118           PAYMENT_DATA orig_data;
1119           char ExtraDebugBuf[2048]={0};
1120
1121           memcpy(&orig_data, pPayment,
  sizeof(orig_data));
1122
1123           sprintf(ExtraDebugBuf, "payment begins w_id
  = %d, c_w_id=%d, h_amout=%.2lf, d_id=%d, c_d_id=%d,
  c_id=%d c_last=%s\n",
1124                       orig_data.w_id,
1125                       orig_data.c_w_id,
1126                       orig_data.h_amount,
1127                       orig_data.d_id,
1128                       orig_data.c_d_id,
1129                       orig_data.c_id,
1130
  orig_data.c_id?"":orig_data.c_last);
1131   #endif
1132
1133           if ( (pEcbInfo = (PECBINFO)dbgetuser-
  data(dbproc)) )
1134           {
1135               pEcbInfo->pECB = pECB;
1136               pEcbInfo->bFailed = FALSE;
1137               pEcbInfo->iTermId = iTermId;
1138               pEcbInfo->iSyncId = iSyncId;
1139           }
1140
1141           pPayment->num_deadlocks = 0;
1142
1143           for (tryit=0; tryit < deadlock_retry;
  tryit++)
1144           {
1145              if (dbrpcinit(dbproc, "tpcc_payment", 0)
  == SUCCEED)
1146               {
1147                  dbrpcparam(dbproc, NULL, 0, SQLINT2,
  -1, -1, (BYTE *) &pPayment->w_id);
1148                  dbrpcparam(dbproc, NULL, 0, SQLINT2,
  -1, -1, (BYTE *) &pPayment->c_w_id);
1149                  dbrpcparam(dbproc, NULL, 0, SQLFLT8,
  -1, -1, (BYTE *) &pPayment->h_amount);
1150                  dbrpcparam(dbproc, NULL, 0, SQLINT1,
  -1, -1, (BYTE *) &pPayment->d_id);
1151                  dbrpcparam(dbproc, NULL, 0, SQLINT1,
  -1, -1, (BYTE *) &pPayment->c_d_id);
1152                  dbrpcparam(dbproc, NULL, 0, SQLINT4,
  -1, -1, (BYTE *) &pPayment->c_id);
1153                   if (pPayment->c_id == 0)
1154                   {
1155                       dbrpcparam(dbproc, NULL, 0, SQL-
  CHAR, -1, strlen(pPayment->c_last), pPayment->c_last);
1156                   }
1157               }
1158   #ifdef EXTRA_DEBUG
1159               else
1160                  sprintf(ExtraDebugBuf+strlen(ExtraDe-
  bugBuf), "\tdbrpcinit failed\n");
1161   #endif
1162               if (dbrpcexec(dbproc) == SUCCEED)
1163               {
1164   #ifdef EXTRA_DEBUG
1165                   sprintf(ExtraDebugBuf+strlen(ExtraDe-
  bugBuf), "\tdbrpcexec == SUCCEED");
1166   #endif
1167                   while (((rc = dbresults(dbproc)) !=
  NO_MORE_RESULTS) && (rc != FAIL))
1168                   {
1169                       if (DBROWS(dbproc) && (dbnum-
  cols(dbproc) == 27))
1170                       {
1171   #ifdef EXTRA_DEBUG
1172                           sprintf(ExtraDebug-
  Buf+strlen(ExtraDebugBuf), " dbnumcols=%d", dbnum-
  cols(dbproc));
1173   #endif
1174                           while (((rc = dbnex-
  trow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
1175                           {
1176                               if(pData=dbdata(dbproc,
  1))
1177                                   pPayment->c_id =
  *((DBINT *) pData);
1178                               if(pData=dbdata(dbproc,
  2))
1179                                   UtilStrCpy(pPayment-
  >c_last, pData, dbdatlen(dbproc, 2));
1180                               if(pData=dbdata(dbproc,
  3))
1181                               {
1182                                   datetime = *((DBDA-
  TETIME *) pData);
1183                                   dbdatecrack(dbproc,
  &pPayment->h_date, &datetime);
1184                               }
1185                               if(pData=dbdata(dbproc,
  4))
1186                                   UtilStrCpy(pPayment-
  >w_street_1, pData, dbdatlen(dbproc, 4));
1187                               if(pData=dbdata(dbproc,
  5))
1188                                   UtilStrCpy(pPayment-
  >w_street_2, pData, dbdatlen(dbproc, 5));
1189                               if(pData=dbdata(dbproc,
  6))
1190                                   UtilStrCpy(pPayment-
  >w_city, pData, dbdatlen(dbproc, 6));
1191                               if(pData=dbdata(dbproc,
  7))
1192                                   UtilStrCpy(pPayment-
  >w_state, pData, dbdatlen(dbproc, 7));
1193                               if(pData=dbdata(dbproc,
  8))
1194                                   UtilStrCpy(pPayment-
  >w_zip, pData, dbdatlen(dbproc, 8));
1195                               if(pData=dbdata(dbproc,
  9))
1196                                   UtilStrCpy(pPayment-
  >d_street_1, pData, dbdatlen(dbproc, 9));
1197                               if(pData=dbdata(dbproc,
  10))
1198                                   UtilStrCpy(pPayment-
  >d_street_2, pData, dbdatlen(dbproc, 10));
1199                               if(pData=dbdata(dbproc,
  11))
1200                                   UtilStrCpy(pPayment-
  >d_city, pData, dbdatlen(dbproc, 11));
1201                               if(pData=dbdata(dbproc,
  12))
1202                                   UtilStrCpy(pPayment-
  >d_state, pData, dbdatlen(dbproc, 12));
1203                               if(pData=dbdata(dbproc,
  13))
1204                                   UtilStrCpy(pPayment-
  >d_zip, pData, dbdatlen(dbproc, 13));
1205                               if(pData=dbdata(dbproc,
```

```
  14))
1206                                    UtilStrCpy(pPayment-
  >c_first, pData, dbdatlen(dbproc, 14));
1207                                if(pData=dbdata(dbproc,
  15))
1208                                    UtilStrCpy(pPayment-
  >c_middle, pData, dbdatlen(dbproc, 15));
1209                                if(pData=dbdata(dbproc,
  16))
1210                                    UtilStrCpy(pPayment-
  >c_street_1, pData, dbdatlen(dbproc, 16));
1211                                if(pData=dbdata(dbproc,
  17))
1212                                    UtilStrCpy(pPayment-
  >c_street_2, pData, dbdatlen(dbproc, 17));
1213                                if(pData=dbdata(dbproc,
  18))
1214                                    UtilStrCpy(pPayment-
  >c_city, pData, dbdatlen(dbproc, 18));
1215                                if(pData=dbdata(dbproc,
  19))
1216                                    UtilStrCpy(pPayment-
  >c_state, pData, dbdatlen(dbproc, 19));
1217                                if(pData=dbdata(dbproc,
  20))
1218                                    UtilStrCpy(pPayment-
  >c_zip, pData, dbdatlen(dbproc, 20));
1219                                if(pData=dbdata(dbproc,
  21))
1220                                    UtilStrCpy(pPayment-
  >c_phone, pData, dbdatlen(dbproc, 21));
1221                                if(pData=dbdata(dbproc,
  22))
1222                                 {
1223                                     datetime = *((DBDA-
  TETIME *) pData);
1224                                    dbdatecrack(dbproc,
  &pPayment->c_since, &datetime);
1225                                 }
1226                                if(pData=dbdata(dbproc,
  23))
1227                                    UtilStrCpy(pPayment-
  >c_credit, pData, dbdatlen(dbproc, 23));
1228                                if(pData=dbdata(dbproc,
  24))
1229                                  pPayment->c_credit_lim
  = (*(DBFLT8 *) pData);
1230                                if(pData=dbdata(dbproc,
  25))
1231                                   pPayment->c_discount
  = (*(DBFLT8 *) pData);
1232                                if(pData=dbdata(dbproc,
  26))
1233                                    pPayment->c_balance
  = (*(DBFLT8 *) pData);
1234                                if(pData=dbdata(dbproc,
  27))
1235                                    UtilStrCpy(pPayment-
  >c_data, pData, dbdatlen(dbproc, 27));
1236                                }
1237                            }
1238                        }
1239   #ifdef EXTRA_DEBUG
1240                    sprintf(ExtraDebugBuf+strlen(ExtraDe-
  bugBuf), " while ends, rc=%d (%s), status=%d\n", rc,
  rc==NO_MORE_ROWS?"NO_MORE_ROWS":(rc==FAIL?"FAIL":"UNKNOW
  N"),
1241                            dbretstatus(dbproc));
1242   #endif
1243                }
1244                else
1245                {
1246   #ifdef EXTRA_DEBUG
1247
1248                    sprintf(ExtraDebugBuf+strlen(ExtraDe-
  bugBuf), "\tdbrpcexed != SUCCEED\n");
1249                    sprintf(ExtraDebugBuf+strlen(ExtraDe-
  bugBuf),
1250                            "\tw_id = %d, c_w_id=%d,
  h_amount=%.2lf, d_id=%d, c_d_id=%d, c_id=%d\n",
1251                                pPayment->w_id,
1252                                pPayment->c_w_id,
1253                                pPayment->h_amount,
1254                                pPayment->d_id,
1255                                pPayment->c_d_id,
1256                                pPayment->c_id);
1257
1258                    sprintf(pPayment->execution_status,
  "dbrpcexec != SUCCEED, w_id = %d, c_w_id=%d, h_amout=%lf,
  d_id=%d, c_d_id=%d, c_id=%d\n",
1259                                orig_data.w_id,
1260                                orig_data.c_w_id,
1261                                orig_data.h_amount,
1262                                orig_data.d_id,
1263                                orig_data.c_d_id,
1264                                orig_data.c_id);
1265                    sprintf(pPayment-
  >execution_status+strlen(pPayment->execution_status),
1266                                "                now:w_id =
  %d, c_w_id=%d, h_amout=%lf, d_id=%d, c_d_id=%d,
  c_id=%d\n",
1267                                pPayment->w_id,
1268                                pPayment->c_w_id,
1269                                pPayment->h_amount,
1270                                pPayment->d_id,
1271                                pPayment->c_d_id,
1272                                pPayment->c_id);
1273                    {
1274                       FILE *fp = fopen(szErrorLogPath,
  "ab");
1275
1276                        EnterCriticalSection(&Error-
  LogCriticalSection);
1277                        fprintf(fp, "%s\n", pPayment-
  >execution_status);
1278                        LeaveCriticalSection(&Error-
  LogCriticalSection);
1279
1280                        fclose(fp);
1281                    }
1282   #else
1283                    strcpy(pPayment-
  >execution_status,"dbrpcexec(dbproc) != SUCCEED");
1284   #endif
1285                }
1286
1287                if (SQLDetectDeadlock(dbproc))
1288                {
1289                    pPayment->num_deadlocks++;
1290                    sprintf(printbuf,"deadlock: retry:
  %d",pPayment->num_deadlocks);
1291                    Sleep(DEADLOCKWAIT*tryit);
1292                }
1293                else
1294                {
1295   #ifdef EXTRA_DEBUG
1296                    EnterCriticalSection(&ErrorLogCrit-
  icalSection);
1297                    fputs(ExtraDebugBuf, stderr);
1298                    LeaveCriticalSection(&ErrorLogCrit-
  icalSection);
1299   #endif
1300                    if ( pPayment->c_id == 0 )
1301                    {
1302                        strcpy(pPayment-
  >execution_status,"Invalid Customer id,name.");
1303                        return 0;
1304                    }
1305                    else
1306                        strcpy(pPayment-
  >execution_status,"Transaction commited.");
1307                    return TRUE;
1308                }
1309            }
```

```
1310
1311   #ifdef EXTRA_DEBUG
1312           EnterCriticalSection(&ErrorLogCriticalSec-
  tion);
1313           fputs(ExtraDebugBuf, stderr);
1314           LeaveCriticalSection(&ErrorLogCriticalSec-
  tion);
1315   #endif
1316         // If we reached here, it means we quit after
  MAX_RETRY deadlocks
1317          strcpy(pPayment->execution_status,"Hit dead-
  lock max.  ");
1318           return -1; //"deadlock max retry reached!"
1319       }
1320   #endif
1321
1322   /* FUNCTION: int (EXTENSION_CONTROL_BLOCK *pECB,
  int iTermId, int iSyncId, DBPROCESS *dbproc,
  ORDER_STATUS_DATA *pOrderStatus, short deadlock_retry)
1323    *
1324    * PURPOSE: This function processes the Order Status
  transaction.
1325    *
1326    * ARGUMENTS:   EXTENSION_CONTROL_BLOCK
  *pECB           passed in structure pointer from inetsrv.
1327    *              int                    iTer-
  mId         terminal id of browser
1328    *              int                    iSyn-
  cId         sync id of browser
1329    *              DBPROCESS
  *dbproc         connection db process id
1330    *           ORDER_STATUS_DATA      *pOrderStatus
  pointer to Order Status data input/output structure
1331    *            short                deadlock_retry
  deadlock retry count
1332    *
1333    * RETURNS:     int -1      max deadlock reached
1334    *                   0      No orders found for cus-
  tomer
1335    *                   1      Transaction successfull
1336    *
1337    * COMMENTS:    None
1338    *
1339    */
1340
1341   #ifdef USE_ODBC
1342       int SQLOrderStatus(EXTENSION_CONTROL_BLOCK
  *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc,
  ORDER_STATUS_DATA *pOrderStatus, short deadlock_retry)
1343       {
1344           int                tryit;
1345           int                i;
1346           BOOL               not_done;
1347           char               buffer[255];
1348           PECBINFO    pEcbInfo;
1349
1350           if ( (pEcbInfo = (PECBINFO)dbgetuser-
  data(dbproc)) )
1351           {
1352               pEcbInfo->pECB = pECB;
1353               pEcbInfo->bFailed = FALSE;
1354               pEcbInfo->iTermId = iTermId;
1355               pEcbInfo->iSyncId = iSyncId;
1356           }
1357
1358           if ( ReopenConnection(dbproc) )
1359               return -3;
1360
1361           pOrderStatus->num_deadlocks = 0;
1362
1363           for (tryit=0; tryit < deadlock_retry;
  tryit++)
1364           {
1365
1366               pEcbInfo->bDeadlock = FALSE;
1367
1368               strcpy(buffer,"{call
  tpcc_orderstatus(?,?,?");
1369               if (pOrderStatus->c_id == 0)
1370                   strcat(buffer,",?");
1371               strcat(buffer,")}");
1372
1373               BindParameter(dbproc, 1, SQL_C_SSHORT,
  SQL_SMALLINT, 0, 0, &pOrderStatus->w_id, 0);
1374              BindParameter(dbproc, 2, SQL_C_STINYINT,
  SQL_TINYINT, 0, 0, &pOrderStatus->d_id, 0);
1375               BindParameter(dbproc, 3, SQL_C_SLONG,
  SQL_INTEGER, 0, 0, &pOrderStatus->c_id, 0);
1376               if (pOrderStatus->c_id == 0)
1377                  BindParameter(dbproc, 4, SQL_C_CHAR,
  SQL_CHAR, (UINT)SQL_NTS, 0, &pOrderStatus->c_last,
  sizeof(pOrderStatus->c_last));
1378
1379               if ( ExecuteStatement(dbproc, buffer) )
1380                   if ( !SQLDetectDeadlock(dbproc) )
1381                       return -2;
1382
1383               not_done = TRUE;
1384               i=0;
1385
1386              while ( not_done && !pEcbInfo->bDeadlock
  )
1387               {
1388                   if ( BindColumn(dbproc, 1,
  SQL_C_SSHORT, &pOrderStatus->OlOrderStatus-
  Data[i].ol_supply_w_id, 0) )
1389                           return -2;
1390                   if ( BindColumn(dbproc, 2,
  SQL_C_SLONG, &pOrderStatus->OlOrderStatus-
  Data[i].ol_i_id, 0) )
1391                           return -2;
1392                   if ( BindColumn(dbproc, 3,
  SQL_C_SSHORT, &pOrderStatus->OlOrderStatus-
  Data[i].ol_quantity, 0) )
1393                           return -2;
1394                   if ( BindColumn(dbproc, 4,
  SQL_C_DOUBLE, &pOrderStatus->OlOrderStatus-
  Data[i].ol_amount, 0) )
1395                           return -2;
1396                   if ( BindColumn(dbproc, 5,
  SQL_C_TIMESTAMP, &pOrderStatus->OlOrderStatus-
  Data[i].ol_delivery_d, 0) )
1397                           return -2;
1398
1399                   switch( SQLFetch(dbproc->hstmt) )
1400                   {
1401                       case SQL_ERROR:
1402                           if ( !pEcbInfo->bDeadlock )
1403                               return -2;
1404                           break;
1405                       case SQL_NO_DATA_FOUND:
1406                           not_done = FALSE;
1407                           break;
1408                       default:
1409                           i++;
1410                           break;
1411                   }
1412               }
1413
1414               pOrderStatus->o_ol_cnt = i;
1415
1416               if ( i )
1417               {
1418                   if ( !pEcbInfo->bDeadlock )
1419                   {
1420                       if ( MoreResults(dbproc) )
1421                       {
1422                           if ( !pEcbInfo->bDeadlock )
1423                               return -2;
1424                       }
1425                       else
1426                       {
1427                           if ( !pEcbInfo->bDeadlock )
1428                           {
```

```
1429                                if ( BindColumn(dbproc,
  1, SQL_C_SLONG, &pOrderStatus->c_id, 0) )
1430                                    return -2;
1431                                if ( BindColumn(dbproc,
  2, SQL_C_CHAR, &pOrderStatus->c_last, sizeof(pOrderSta-
  tus->c_last)) )
1432                                    return -2;
1433                                if ( BindColumn(dbproc,
  3, SQL_C_CHAR, &pOrderStatus->c_first, sizeof(pOrderSta-
  tus->c_first)) )
1434                                    return -2;
1435                                if ( BindColumn(dbproc,
  4, SQL_C_CHAR, &pOrderStatus->c_middle, sizeof(pOrder-
  Status->c_middle)) )
1436                                    return -2;
1437                                if ( BindColumn(dbproc,
  5, SQL_C_TIMESTAMP, &pOrderStatus->o_entry_d, 0) )
1438                                    return -2;
1439                                if ( BindColumn(dbproc,
  6, SQL_C_SSHORT, &pOrderStatus->o_carrier_id, 0) )
1440                                    return -2;
1441                                if ( BindColumn(dbproc,
  7, SQL_C_DOUBLE, &pOrderStatus->c_balance, 0) )
1442                                    return -2;
1443                                if ( BindColumn(dbproc,
  8, SQL_C_SLONG, &pOrderStatus->o_id, 0) )
1444                                    return -2;
1445
1446                               if ( GetResults(dbproc) )
1447                                    return -2;
1448                           }
1449                       }
1450                   }
1451               }
1452               else
1453               {
1454                   SQLFreeStmt(dbproc->hstmt,
  SQL_CLOSE);
1455                  return 0; //"No orders found for cus-
  tomer"
1456               }
1457
1458               SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
1459
1460               if ( pEcbInfo->bDeadlock )
1461               {
1462                   pOrderStatus->num_deadlocks++;
1463                   Sleep(DEADLOCKWAIT*tryit);
1464               }
1465               else
1466               {
1467                   if (pOrderStatus->c_id == 0 &&
  pOrderStatus->c_last[0] == 0)
1468                       strcpy(pOrderStatus-
  >execution_status,"Invalid Customer id,name.");
1469                   else
1470                       strcpy(pOrderStatus-
  >execution_status,"Transaction commited.");
1471                   return 1;
1472               }
1473           }
1474
1475         // If we reached here, it means we quit after
  MAX_RETRY deadlocks
1476           strcpy(pOrderStatus->execution_status,"Hit
  deadlock max.  ");
1477           return -1;
1478       }
1479   #else
1480       int SQLOrderStatus(EXTENSION_CONTROL_BLOCK
  *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc,
  ORDER_STATUS_DATA *pOrderStatus, short deadlock_retry)
1481       {
1482           RETCODE     rc;
1483           int         tryit;
1484           int         i;
1485           char        printbuf[25];
1486           DBDATETIME  datetime;
1487           BYTE        *pData;
1488           PECBINFO    pEcbInfo;
1489
1490           if ( (pEcbInfo = (PECBINFO)dbgetuser-
  data(dbproc)) )
1491           {
1492               pEcbInfo->pECB = pECB;
1493               pEcbInfo->bFailed = FALSE;
1494               pEcbInfo->iTermId = iTermId;
1495               pEcbInfo->iSyncId = iSyncId;
1496           }
1497
1498           pOrderStatus->num_deadlocks = 0;
1499
1500           for (tryit=0; tryit < deadlock_retry;
  tryit++)
1501           {
1502               if (dbrpcinit(dbproc,
  "tpcc_orderstatus", 0) == SUCCEED)
1503               {
1504                   dbrpcparam(dbproc, NULL, 0, SQLINT2,
  -1, -1, (BYTE *) &pOrderStatus->w_id);
1505                   dbrpcparam(dbproc, NULL, 0, SQLINT1,
  -1, -1, (BYTE *) &pOrderStatus->d_id);
1506                   dbrpcparam(dbproc, NULL, 0, SQLINT4,
  -1, -1, (BYTE *) &pOrderStatus->c_id);
1507                    if (pOrderStatus->c_id == 0)
1508                    {
1509                       dbrpcparam(dbproc, NULL, 0, SQL-
  CHAR, -1, strlen(pOrderStatus->c_last), pOrderStatus-
  >c_last);
1510                    }
1511               }
1512               if (dbrpcexec(dbproc) == SUCCEED)
1513               {
1514                   while (((rc = dbresults(dbproc)) !=
  NO_MORE_RESULTS) && (rc != FAIL))
1515                   {
1516                       if (DBROWS(dbproc) && (dbnum-
  cols(dbproc) == 5))
1517                       {
1518                           i=0;
1519                           while (((rc = dbnex-
  trow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
1520                           {
1521                               if(pData=dbdata(dbproc,
  1))
1522                                 pOrderStatus->OlOrder-
  StatusData[i].ol_supply_w_id = (*(DBSMALLINT *) pData);
1523                               if(pData=dbdata(dbproc,
  2))
1524                                 pOrderStatus->OlOrder-
  StatusData[i].ol_i_id = (*(DBINT *) pData);
1525                               if(pData=dbdata(dbproc,
  3))
1526                                 pOrderStatus->OlOrder-
  StatusData[i].ol_quantity = (*(DBSMALLINT *) pData);
1527                               if(pData=dbdata(dbproc,
  4))
1528                                 pOrderStatus->OlOrder-
  StatusData[i].ol_amount = (*(DBFLT8 *) pData);
1529                               if(pData=dbdata(dbproc,
  5))
1530                                {
1531                                    datetime = *((DBDA-
  TETIME *) pData);
1532                                   dbdatecrack(dbproc,
  &pOrderStatus->OlOrderStatusData[i].ol_delivery_d,
  &datetime);
1533                                }
1534                                i++;
1535                           }
1536                           pOrderStatus->o_ol_cnt = i;
1537                       }
1538                       else if (DBROWS(dbproc) &&
  (dbnumcols(dbproc) == 8))
```

```
1539                        {
1540                            while (((rc = dbnex-
  trow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
1541                            {
1542                                if(pData=dbdata(dbproc,
  1))
1543                                   pOrderStatus->c_id =
  (*(DBINT *) pData);
1544                                if(pData=dbdata(dbproc,
  2))
1545                                    UtilStrCpy(pOrder-
  Status->c_last, pData, dbdatlen(dbproc,2));
1546                                if(pData=dbdata(dbproc,
  3))
1547                                    UtilStrCpy(pOrder-
  Status->c_first, pData, dbdatlen(dbproc,3));
1548                                if(pData=dbdata(dbproc,
  4))
1549                                    UtilStrCpy(pOrder-
  Status->c_middle, pData, dbdatlen(dbproc, 4));
1550                                if(pData=dbdata(dbproc,
  5))
1551                                 {
1552                                     datetime = *((DBDA-
  TETIME *) pData);
1553                                    dbdatecrack(dbproc,
  &pOrderStatus->o_entry_d, &datetime);
1554                                 }
1555                                if(pData=dbdata(dbproc,
  6))
1556                                    pOrderStatus-
  >o_carrier_id = (*(DBSMALLINT *) pData);
1557                                if(pData=dbdata(dbproc,
  7))
1558                                    pOrderStatus-
  >c_balance = (*(DBFLT8 *) pData);
1559                                if(pData=dbdata(dbproc,
  8))
1560                                   pOrderStatus->o_id =
  (*(DBINT *) pData);
1561                            }
1562                        }
1563                        if (i==0)
1564                           return 0; //"No orders found
  for customer"
1565                    }
1566                }
1567                if (SQLDetectDeadlock(dbproc))
1568                {
1569                    pOrderStatus->num_deadlocks++;
1570                    sprintf(printbuf,"deadlock: retry:
  %d",pOrderStatus->num_deadlocks);
1571                    Sleep(DEADLOCKWAIT*tryit);
1572                }
1573                else
1574                {
1575                    if (pOrderStatus->c_id == 0 &&
  pOrderStatus->c_last[0] == 0)
1576                        strcpy(pOrderStatus-
  >execution_status,"Invalid Customer id,name.");
1577                    else
1578                        strcpy(pOrderStatus-
  >execution_status,"Transaction commited.");
1579                    return 1;
1580                }
1581            }
1582          // If we reached here, it means we quit after
  MAX_RETRY deadlocks
1583            strcpy(pOrderStatus->execution_status,"Hit
  deadlock max.  ");
1584            return -1; //"deadlock max retry reached!"
1585        }
1586   #endif
1587
1588   /* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS
  *dbproc)
1589    *
1590    * PURPOSE: This function checks to see if a sql
  server deadlock condition exists.
1591    *
1592    * ARGUMENTS:   DBPROCESS           *dbproc
  connection db process id to check
1593    *
1594    * RETURNS:     BOOL    FALSE       no deadlock
  detected
1595    *                     TRUE        deadlock condition
  exists
1596    *
1597    * COMMENTS:    None
1598    *
1599    */
1600
1601   BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
1602   {
1603       PECBINFO    pEcbInfo;
1604
1605       if ( (pEcbInfo = (PECBINFO)dbgetuser-
  data(dbproc)) )
1606       {
1607           if ( pEcbInfo->bDeadlock )
1608           {
1609               pEcbInfo->bDeadlock = FALSE;
1610               return TRUE;
1611           }
1612       }
1613       return FALSE;
1614   }
1615
1616
1617   #ifdef USE_ODBC
1618
1619       /* FUNCTION: void dbsetuserdata(PDBPROCESS
  dbproc, void *uPtr)
1620        *
1621        * PURPOSE: This function sets a user pointer in
  a dbproc structure
1622        *          This functionality is not provided
  in odbc so this function
1623        *          provides it.
1624        *
1625        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1626        *              void        *uPtr   returned data
  user pointer
1627        *
1628        * RETURNS:     none
1629        *
1630        * COMMENTS:    The caller is responsible for
  the contents of the uPtr.
1631        *
1632        */
1633
1634       void dbsetuserdata(PDBPROCESS dbproc, void
  *uPtr)
1635       {
1636           dbproc->uPtr = uPtr;
1637       }
1638
1639       /* FUNCTION: void dbsetuserdata(PDBPROCESS
  dbproc, void *uPtr)
1640        *
1641        * PURPOSE: This function returns the user
  pointer stored in a dbproc structure
1642        *          This functionality is not provided
  in odbc so this function
1643        *          provides it.
1644        *
1645        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1646        *
1647        * RETURNS:     none
1648        *
1649        * COMMENTS:    The returned pointer is placed
  in the dbproc structure by the dbsetuserdata() API.
```

```
1650        *
1651        */
1652
1653       void *dbgetuserdata(PDBPROCESS dbproc)
1654       {
1655           return dbproc->uPtr;
1656       }
1657
1658       /* FUNCTION: void BindParameter(PDBPROCESS
  dbproc, UWORD ipar, SWORD fCType, SWORD fSqlType, UDWORD
  cbColDef, SWORD ibScale, PTR rgbValue, SDWORD cbValueMax)
1659        *
1660        * PURPOSE: This function wraps the functional-
  ity provided by the SQLBindParameter
1661        *          allowing error process so that each
  bind call does not need to provide
1662        *          error and message checking.
1663        *
1664        * ARGUMENTS:   PDBPROCESS  dbproc      pointer
  to odbc dbprocess structure
1665        *              UWORD   ipar        Parameter
  number, ordered sequentially left to right, starting at
  1.
1666        *              SWORD   fParamType  The type of
  the parameter.
1667        *              SWORD   fCType      The C data
  type of the parameter.
1668        *              SWORD   fSqlType    The SQL data
  type of the parameter.
1669        *             UDWORD  cbColDef    The precision
  of the column or expression
1670        *                                   of the cor-
  responding parameter marker.
1671        *              SWORD   ibScale     The scale of
  the column or expression of the corresponding
1672        *                                   parameter
  marker.
1673        *              PTR     rgbValue    A pointer to
  a buffer for the parameters data.
1674        *              SDWORD  cbValueMax  Maximum
  length of the rgbValue buffer.
1675        *             void       *uPtr   returned data
  user pointer
1676        *
1677        * RETURNS:     none
1678        *
1679        * COMMENTS:    The returned pointer is placed
  in the dbproc structure by the dbset
1680        *
1681        */
1682
1683       void BindParameter(PDBPROCESS dbproc, UWORD
  ipar, SWORD fCType, SWORD fSqlType, UDWORD cbColDef,
  SWORD ibScale, PTR rgbValue, SDWORD cbValueMax)
1684       {
1685           RETCODE rc;
1686
1687           if ( ((PECBINFO)dbgetuserdata(dbproc))-
  >bFailed )
1688               return;
1689           rc = SQLBindParameter(dbproc->hstmt, ipar,
  SQL_PARAM_INPUT, fCType, fSqlType, cbColDef, ibScale,
  rgbValue, cbValueMax, NULL);
1690           if (rc == SQL_ERROR)
1691               ODBCError(dbproc);
1692           return;
1693       }
1694
1695       /* FUNCTION: void ODBCError(PDBPROCESS dbproc)
1696        *
1697        * PURPOSE: This function wraps the odbc error
  call so that the dblib msg_handler is called.
1698        *         This allows the deadlock flag in the
  dbproc user data structure pEcbInfo in
1699        *          dbproc to be set if necessary.
1700        *
1701        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1702        *
1703        * RETURNS:     none
1704        *
1705        * COMMENTS:    none
1706        *
1707        */
1708
1709       void ODBCError(PDBPROCESS dbproc)
1710       {
1711           SDWORD      lNativeError;
1712           char        szState[6];
1713           char        szMsg[SQL_MAX_MESSAGE_LENGTH];
1714           char        szMsgText[256];
1715           PECBINFO    pEcbInfo;
1716           char        szTmp[256];
1717           FILE        *fp;
1718           SYSTEMTIME  systemTime;
1719
1720           pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
1721           while( SQLError(henv, dbproc->hdbc, dbproc-
  >hstmt, szState, &lNativeError, szMsg, sizeof(szMsg),
  NULL) == SQL_SUCCESS )
1722           {
1723               msg_handler(dbproc, lNativeError, 0, 0,
  szMsg);
1724               if ( !lNativeError )
1725               {
1726                   sprintf(szMsgText, "State = %s, %s",
  szState, szMsg);
1727                   ErrorMessage(pEcbInfo->pECB, -1,
  ERR_TYPE_ODBC, szMsgText, pEcbInfo->iTermId, pEcbInfo-
  >iSyncId);
1728                   pEcbInfo->bFailed = TRUE;
1729
1730                   GetLocalTime(&systemTime);
1731                   fp = fopen(szErrorLogPath, "ab");
1732
1733                   EnterCriticalSection(&ErrorLogCrit-
  icalSection);
1734                  sprintf(szTmp, "Error: SQLSVR(): %s",
  szMsg);
1735                   fprintf(fp, "%2.2d/%2.2d/%2.2d
  %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
1736                           systemTime.wYear, system-
  Time.wMonth, systemTime.wDay,
1737                           systemTime.wHour, system-
  Time.wMinute, systemTime.wSecond,
1738                           szTmp);
1739                   LeaveCriticalSection(&ErrorLogCrit-
  icalSection);
1740
1741                   fclose(fp);
1742               }
1743           }
1744
1745           return;
1746       }
1747
1748       /* FUNCTION: BOOL ExecuteStatement(PDBPROCESS
  dbproc, szStatement)
1749        *
1750        * PURPOSE: This function wraps the odbc SQLEx-
  ecDirect API so that error handling and
1751        *          and deadlock are taken care of in a
  common location.
1752        *
1753        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1754        *              char        *szStatement    sql
  stored procedure statement to be executed.
1755        *
1756        * RETURNS:     none
1757        *
1758        * COMMENTS:    none
1759        *
1760        */
```

```
1761
1762       BOOL ExecuteStatement(PDBPROCESS dbproc, char
  *szStatement)
1763       {
1764           RETCODE     rc;
1765           PECBINFO    pEcbInfo;
1766
1767           pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
1768           if ( pEcbInfo->bFailed )
1769               return TRUE;
1770
1771           rc = SQLExecDirect(dbproc->hstmt, szState-
  ment, SQL_NTS);
1772           if (rc != SQL_SUCCESS && rc !=
  SQL_SUCCESS_WITH_INFO)
1773           {
1774               ODBCError(dbproc);
1775               if ( pEcbInfo->bDeadlock )
1776                   return FALSE;
1777               return TRUE;
1778           }
1779           return FALSE;
1780       }
1781
1782       /* FUNCTION: BOOL BindColumn(PDBPROCESS dbproc,
  SQLUSMALLINT icol, SQLSMALLINT fCType, SQLPOINTER rgb-
  Value, SQLINTEGER cbValueMax)
1783        *
1784        * PURPOSE: This function wraps the odbc SQL-
  BindCol API so that error handling and
1785        *          and deadlock are taken care of in a
  common location.
1786        *
1787        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1788        *              UWORD   icol            Column
  number of result data, ordered sequentially left to
  right, starting at 1.
1789        *              SWORD   fCType          The C
  data type of the result data. SQL_C_BINARY, SQL_C_BIT,
  SQL_C_BOOKMARK,
1790        *                                      SQL_C_CHAR,
  SQL_C_DATE, SQL_C_DEFAULT, SQL_C_DOUBLE, SQL_C_FLOAT,
  SQL_C_SLONG,
1791        *
  SQL_C_SSHORT, SQL_C_STINYINT, SQL_C_TIME,
  SQL_C_TIMESTAMP, SQL_C_ULONG,
1792        *
  SQL_C_USHORT, SQL_C_UTINYINT, SQL_C_DEFAULT
1793        *              PTR     rgbValue        Pointer
  to storage for the data. If rgbValue is a null pointer,
  the
1794        *                                      driver
  unbinds the column.
1795        *              SDWORD  cbValueMax      Maximum
  length of the rgbValue buffer. For character data, rgb-
  Value
1796        *                                      must also
  include space for the null-termination byte.
1797        * RETURNS:     none
1798        *
1799        * COMMENTS:    none
1800        *
1801        */
1802
1803       BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT
  icol, SQLSMALLINT fCType, SQLPOINTER rgbValue, SQLINTEGER
  cbValueMax)
1804       {
1805           RETCODE     rc;
1806           PECBINFO    pEcbInfo;
1807
1808           pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
1809           if ( pEcbInfo->bFailed )
1810               return TRUE;
1811
1812          rc = SQLBindCol(dbproc->hstmt, icol, fCType,
  rgbValue, cbValueMax, NULL);
1813           if ( rc == SQL_ERROR )
1814           {
1815               ODBCError(dbproc);
1816               return TRUE;
1817           }
1818           return FALSE;
1819       }
1820
1821       /* FUNCTION: BOOL GetResults(PDBPROCESS dbproc)
1822        *
1823        * PURPOSE: This function wraps the odbc
  SQLFetch API so that error handling and
1824        *          and deadlock are taken care of in a
  common location.
1825        *
1826        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1827        *
1828        * RETURNS:     none
1829        *
1830        * COMMENTS:    none
1831        *
1832        */
1833
1834       BOOL GetResults(PDBPROCESS dbproc)
1835       {
1836           PECBINFO pEcbInfo;
1837
1838           pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
1839           if ( pEcbInfo->bFailed )
1840               return TRUE;
1841
1842           if ( SQLFetch(dbproc->hstmt) == SQL_ERROR )
1843           {
1844               ODBCError(dbproc);
1845               if ( pEcbInfo->bDeadlock )
1846                   return FALSE;
1847               return TRUE;
1848           }
1849           return FALSE;
1850       }
1851
1852       /* FUNCTION: BOOL MoreResults(DBPROCESS  dbproc)
1853        *
1854        * PURPOSE: This function wraps the odbc SQLMor-
  eResults API so that error handling and
1855        *          and deadlock are taken care of in a
  common location.
1856        *
1857        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1858        *
1859        * RETURNS:     none
1860        *
1861        * COMMENTS:    none
1862        *
1863        */
1864
1865       BOOL MoreResults(PDBPROCESS  dbproc)
1866       {
1867           PECBINFO    pEcbInfo;
1868
1869           pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
1870           if ( pEcbInfo->bFailed )
1871               return TRUE;
1872
1873           if ( SQLMoreResults(dbproc->hstmt) ==
  SQL_ERROR )
1874           {
1875               ODBCError(dbproc);
1876               if ( pEcbInfo->bDeadlock )
1877                   return FALSE;
1878               return TRUE;
1879           }
1880           return FALSE;
1881       }
```

```
1882
1883       /* FUNCTION: BOOL ReopenConnection(PDBPROCESS
  dbproc)
1884        *
1885        * PURPOSE: This function is used with connec-
  tion ODBC pooling to reissue the
1886        *          close hdbc connection.
1887        *
1888        * ARGUMENTS:   DBRPOCESS   dbproc  ODBC dbpro-
  cess structure
1889        *
1890        * RETURNS:     FALSE if successfull
1891        *              TRUE if an error occurs
1892        *
1893        * COMMENTS:    none
1894        *
1895        */
1896
1897       BOOL ReopenConnection(PDBPROCESS dbproc)
1898       {
1899           RETCODE     rc;
1900           PECBINFO    pEcbInfo;
1901           int         iCount;
1902           FILE        *fp;
1903           SYSTEMTIME  systemTime;
1904
1905           if ( !bConnectionPooling )
1906               return FALSE;
1907
1908           pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
1909
1910           iCount = 0;
1911
1912           /*  I don't think this is necessary.  ODBC
  connection pooling should remember this. - damienl
1913
1914           if ( SQLSetConnectOption(dbproc->hdbc,
  SQL_PACKET_SIZE, 4096) == SQL_ERROR )
1915           {
1916               ODBCError(dbproc);
1917               return TRUE;
1918           }
1919           */
1920
1921          if (SQLAllocConnect(henv, &dbproc->hdbc) ==
  SQL_ERROR)
1922           {
1923               ODBCError(dbproc);
1924               return TRUE;
1925           }
1926
1927           rc = SQLConnect(dbproc->hdbc, szServer,
  SQL_NTS, szUser, SQL_NTS, szPassword, SQL_NTS);
1928           while (rc != SQL_SUCCESS && rc !=
  SQL_SUCCESS_WITH_INFO)
1929           {
1930
1931               Sleep(iConnectDelay); //wait and try
  again
1932
1933               iCount++;
1934               if ( (iCount % 1) == 0)
1935               {
1936                   fp = fopen(szErrorLogPath, "ab");
1937
1938                   GetLocalTime(&systemTime);
1939
1940                   fprintf(fp, "* CONNECTION POOL *
  %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d TermId = %d, SyncID =
  %d, Spin Count = %d\r\n\r\n",
1941                           systemTime.wYear, system-
  Time.wMonth, systemTime.wDay,
1942                           systemTime.wHour, system-
  Time.wMinute, systemTime.wSecond,
1943                           pEcbInfo->iTermId, pEcbInfo-
  >iSyncId, iCount);
1944
1945                   fclose(fp);
1946               }
1947
1948              rc = SQLConnect(dbproc->hdbc, szServer,
  SQL_NTS, szUser, SQL_NTS, szPassword, SQL_NTS);
1949           }
1950
1951           rc = SQLAllocStmt(dbproc->hdbc, &dbproc-
  >hstmt);
1952           if (rc == SQL_ERROR)
1953           {
1954               ODBCError(dbproc);
1955               return TRUE;
1956           }
1957
1958           rc = SQLExecDirect((dbproc)->hstmt, "use
  tpcc set nocount on set XACT_ABORT ON", SQL_NTS);
1959           if (rc != SQL_SUCCESS && rc !=
  SQL_SUCCESS_WITH_INFO)
1960           {
1961               ODBCError(dbproc);
1962               return TRUE;
1963           }
1964           SQLFreeStmt((dbproc)->hstmt, SQL_CLOSE);
1965
1966           return FALSE;
1967
1968       }
1969
1970   #endif
1971
1972   PECBINFO SQLGetECB(PDBPROCESS p)
1973   {
1974       return (PECBINFO)dbgetuserdata(p);
1975   }
1976
```

sqlroutines.h

```
1   //this structure allows the EXTENSION CONTROL BLOCK
to be passed to the msg and error handlers.
2   typedef struct _ECBINFO
3   {
4       int                    iTermId;    //terminal id
5       int                     iSyncId;    //browser
sync id
6       BOOL                    bDeadlock;  //deadlock
condition flag
7       BOOL                    bFailed;    //cleared
before sql transaction, set in err handlers if an error
occurs
8       EXTENSION_CONTROL_BLOCK *pECB;      //inetsrv
current connection structure information
9   } ECBINFO, *PECBINFO;
10
11  BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char
*server, char *database, char *user, char *password, char
*app, int *spid);
12  BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK
*pECB, DBPROCESS *dbproc);
13  BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB,
int iTermId, int iSyncId, DBPROCESS *dbproc,
STOCK_LEVEL_DATA *pStockLevel, short deadlock_retry);
14  int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA
*pNewOrder, short deadlock_retry);
15  int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA
*pPayment, short deadlock_retry);
16  int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB,
int iTermId, int iSyncId, DBPROCESS *dbproc,
ORDER_STATUS_DATA *pOrderStatus, short deadlock_retry);
17  BOOL SQLInit(void);
18  void SQLCleanup(void);
19  BOOL SQLThreadAttach(void);
20  BOOL SQLThreadDetach(void);
```

```
   21   PECBINFO SQLGetECB(PDBPROCESS p);
   22
```

tpcc.c

```
    1    /*  FILE:       TPCC.C
    2     *              Microsoft TPC-C Kit Ver. 3.00.000
    3     *             Audited 08/23/96    By Francois Raab
    4     *
    5     *              Copyright Microsoft, 1996
    6     *
    7     *  PURPOSE:    Main module for TPCC.DLL which is an
   ISAPI service dll.
    8     *  Author:     Philip Durr
    9     *              philipdu@Microsoft.com
   10     */
   11
   12    #include <windows.h>
   13    #include <process.h>
   14    #include <stdio.h>
   15    #include <stdarg.h>
   16    #include <malloc.h>
   17    #include <stdlib.h>
   18    #include <string.h>
   19    #include <time.h>
   20    #include <sys\timeb.h>
   21    #include <io.h>
   22    #include <fcntl.h>
   23
   24
   25    #include "trans.h"                      //tpckit
   transaction header contains definations of structures
   specific to TPC-C
   26    #include "httpext.h"                   //ISAPI DLL
   information header
   27
   28    #include "tpcc.h"                  //this dlls spe-
   cific structure, value e.t. header.
   29
   30    #include "sqlroutines.h"       // the header files
   for the SQL routines (may be hiding TUX)
   31    #include "util.h"
   32    #include "error.h"
   33
   34    #ifdef USE_ODBC
   35        HENV    henv;
   36    #endif
   37
   38    char    szServer[32]    = { 0 };        //global
   variables used with this DLL
   39    char    szUser[32]      = { 0 };
   40    char    szPassword[32]  = { 0 };
   41    char    szDatabase[32]  = "tpcc";
   42    BOOL    bLog            = FALSE;
   43    int     iThreads        = 5;
   44    int     iMaxWareHouses  = 500;
   45    int     iQSlotts        = 3000;
   46    int     iDelayMs        = 100;
   47    int     iConnectDelay   = 500;
   48    short   iDeadlockRetry  = (short)3;
   49    short   iMaxConnections = (short)25;
   50
   51    #ifdef USE_ODBC
   52        int     bConnectionPooling = FALSE;
   53    #endif
   54
   55    //allowable client command strings i.e. CMD=command
   56    char *szCmds[] =
   57    {
   58        "..NewOrder..", "..Payment..", "..Delivery..",
   "..Order-Status..", "..Stock-Level..", "..Exit..",
   59        "Submit", "Begin", "Process", "Menu", "Clear",
   "Users", ""
   60    };
   61
   62
   63    //defined command string functions, called via
   CMD=command http string from html client.
   64
   65    void (*DoCmd[])(EXTENSION_CONTROL_BLOCK *pECB, int
   iFormId, int iTermId, int iSyncId) =
   66    {
   67        NewOrderForm,
   68        PaymentForm,
   69        DeliveryForm,
   70        OrderStatusForm,
   71        StockLevelForm,
   72        Exitcmd,
   73        SubmitCmd,
   74        BeginCmd,
   75        ProcessCmd,
   76        MenuCmd,
   77        ClearCmd,
   78        NumberOfConnectionsCmd
   79    };
   80
   81    //Terminal client id structure and interface defi-
   nation
   82    TERM    Term = { 0, 0, 0, FALSE, NULL, TermInit,
   TermAllocate, TermRestore, TermAdd, TermDelete };
   83
   84    //welcome to tpc-c html form buffer, this is first
   form client sees.
   85    static char *szWelcomeForm =    "<HTML>"
   86                                  "<HEAD><TITLE>Welcome
   To TPC-C</TITLE></HEAD><BODY>"
   87                                  "Please Identify your
   Warehouse and District for this session.<BR>"
   88                                    "<FORM
   ACTION=\"tpcc.dll\" METHOD=\"GET\">"
   89                                    "<INPUT TYPE=\"hid-
   den\" NAME=\"STATUSID\" VALUE=\"0\">"
   90                                    "<INPUT TYPE=\"hid-
   den\" NAME=\"FORMID\" VALUE=\"1\">"
   91                                    "<INPUT TYPE=\"hid-
   den\" NAME=\"TERMID\" VALUE=\"-2\">"
   92                                    "<INPUT TYPE=\"hid-
   den\" NAME=\"SYNCID\" VALUE=\"0\">"
   93                                  "Warehouse ID <INPUT
   NAME=\"w_id\" SIZE=4><BR>"
   94                                   "District ID <INPUT
   NAME=\"d_id\" SIZE=2><BR>"
   95                                    "<HR>"
   96                                    "<INPUT TYPE=\"sub-
   mit\" NAME=\"CMD\" VALUE=\"Submit\">"
   97                                    "</FORM><BODY>"
   98                                    "</HTML>";
   99
  100    static char szTpccLogPath[256]; //path to html log
   file if logging turned on in registry.
  101    char szErrorLogPath[256];   //path to error log
   file.
  102
  103    static  CRITICAL_SECTION        CriticalSection;
  104
  105    static  LPTSTR                  lpszPipeName    =
   TEXT("\\\\.\\pipe\\DELISRV");
  106    static  HANDLE                  hDeliveryWrite  =
   INVALID_HANDLE_VALUE;
  107    static  HANDLE                  hPipe           =
   INVALID_HANDLE_VALUE;
  108
  109    EXTENSION_CONTROL_BLOCK *gpECB;
  110    static  int                     bTpccExit;  //exit
   delivery disconnect loop as dll exiting.
  111
  112    /* FUNCTION: BOOL APIENTRY DllMain(HANDLE hModule,
   DWORD ul_reason_for_call, LPVOID lpReserved)
  113     *
  114     * PURPOSE: This function is the entry point for the
   DLL this implementation is baised on the
  115     *          fact that DLL_PROCESS_ATTACH is only
   called from the inet service once. Connections
  116     *          are sent to this function as thread
```

```
 attachments.
117     *
118     * ARGUMENTS:   HANDLE  hModule             module
 handle
119     *              DWORD   ul_reason_for_call  reason
 for call
120     *             LPVOID  lpReserved         reserved
 for future use
121     *
122     * RETURNS:     BOOL    FALSE               errors
 occured in initialization
123     *                     TRUE                DLL suc-
 cessfully initialized
124     *
125     * COMMENTS:    None
126     *
127     */
128
129    BOOL APIENTRY DllMain(HANDLE hModule, DWORD
 ul_reason_for_call, LPVOID lpReserved)
130    {
131        int             i;
132        static SECURITY_ATTRIBUTES  sa;
133        static PSECURITY_DESCRIPTOR pSD;
134
135        switch( ul_reason_for_call )
136        {
137            case DLL_PROCESS_ATTACH:
138                {
139                    const char *Log-
 File="\\temp\\tpcc_logs\\tpcc.dll.txt";
140                    mkpath(LogFile);
141                    freopen(LogFile, "a", stderr);
142                    setbuf(stderr, NULL);
143                    fprintf(stderr, "logging
 started\n");
144                }
145
146                if ( ReadRegistrySettings() )
147                {
148                    MessageBox(NULL, "Cannot Find TPCC
 Key in registry (run install.exe).", "Init", MB_OK |
 MB_ICONSTOP);
149                    return FALSE;
150                }
151
152                InitializeCriticalSection(&CriticalSec-
 tion);
153                (*Term.Init)();
154                if ( !(*Term.Allocate)() )
155                {
156                    MessageBox(NULL, "Error Trm.Allo-
 cate().", "Init", MB_OK | MB_ICONSTOP);
157                    return FALSE;
158                }
159                for(i=Term.iNext; i<Term.iAvailable;
 i++)
160                    Term.pClientData[i].inUse = 0;
161                Term.pClientData[0].inUse = 1;
162
163                // create a security descriptor that
 allows anyone to access the pipe...
164                pSD = (PSECURITY_DESCRIPTOR)malloc(
 SECURITY_DESCRIPTOR_MIN_LENGTH );
165                if ( pSD == NULL )
166                {
167                    MessageBox(NULL, "Error mal-
 loc(SECURITY_DESCRIPTOR_MIN_LENGTH)", "Init", MB_OK |
 MB_ICONSTOP);
168                    return FALSE;
169                }
170               if ( !InitializeSecurityDescriptor(pSD,
 SECURITY_DESCRIPTOR_REVISION) )
171                {
172                    MessageBox(NULL, "Error Initializ-
 eSecurityDescriptor()", "Init", MB_OK | MB_ICONSTOP);
173                    return FALSE;
174                }
175               // add a NULL disc. ACL to the security
 descriptor.
176                if ( !SetSecurityDescriptorDacl(pSD,
 TRUE, (PACL) NULL, FALSE) )
177                {
178                    MessageBox(NULL, "Error SetSecurity-
 DescriptorDacl().", "Init", MB_OK | MB_ICONSTOP);
179                    return FALSE;
180                }
181
182                sa.nLength              = sizeof(sa);
183                sa.lpSecurityDescriptor = pSD;
184                sa.bInheritHandle       = TRUE;
185
186                // open delivery named pipe...
187                hPipe = CreateNamedPipe(lpszPipeName,
 FILE_FLAG_OVERLAPPED | PIPE_ACCESS_DUPLEX,
188                    PIPE_TYPE_BYTE | PIPE_READMODE_BYTE
 | PIPE_NOWAIT,
189                    1, 65535, 65535, 250, &sa);
190
191                if ( hPipe == INVALID_HANDLE_VALUE )
192                {
193                    MessageBox(NULL, "Error CreateNamed-
 Pipe().", "Init", MB_OK | MB_ICONSTOP);
194                    free(pSD);
195                    return FALSE;
196                }
197
198                bTpccExit = FALSE;
199
200                if ( _beginthread( DeliveryDisconnect,
 0, NULL ) == -1 )
201                {
202                    MessageBox(NULL, "Error
 _beginthread()", "Init", MB_OK | MB_ICONSTOP);
203                    return FALSE;
204                }
205
206                if (!SQLInit())
207                        return FALSE;
208                break;
209            case DLL_THREAD_ATTACH:
210                if (!SQLThreadAttach())
211                    return FALSE;
212                break;
213            case DLL_THREAD_DETACH:
214                if (!SQLThreadDetach())
215                    return FALSE;
216                break;
217            case DLL_PROCESS_DETACH:
218                if ( pSD )
219                    free( pSD );
220
221                bTpccExit = TRUE;
222
223                if ( hPipe )
224                    DisconnectNamedPipe(hPipe);
225
226                if (hPipe != INVALID_HANDLE_VALUE )
227                    CloseHandle(hPipe);
228
229                (*Term.Restore)();
230
231                SQLCleanup();
232
233               DeleteCriticalSection(&CriticalSection);
234
235                break;
236        }
237        return TRUE;
238    }
239
240    /* FUNCTION: void DeliveryDisconnect(void *ptr)
241     *
242     * PURPOSE: This function handles disconnecting the
```

```
 server side of the delivery pipe when the
243     *          delivery handler application shuts down.
244     *
245     * ARGUMENTS:   void    *ptr    void pointer nor-
 mally NULL passed from thread handler.
246     *
247     * RETURNS:     None
248     *
249     * COMMENTS:    This function runs as thread which
 allows the client pipe to disconnect by
250     *              sending a byte back though the pipe
 to the server i.e. this DLL.
251     */
252
253    static void DeliveryDisconnect(void *ptr)
254    {
255        int                     l, d;
256        SECURITY_ATTRIBUTES     sa;
257        PSECURITY_DESCRIPTOR    pSD;
258
259        // create a security descriptor that allows any-
 one to access the pipe...
260        pSD = (PSECURITY_DESCRIPTOR)malloc(
 SECURITY_DESCRIPTOR_MIN_LENGTH );
261        InitializeSecurityDescriptor(pSD,
 SECURITY_DESCRIPTOR_REVISION);
262        SetSecurityDescriptorDacl(pSD, TRUE, (PACL)
 NULL, FALSE);
263        sa.nLength              = sizeof(sa);
264        sa.lpSecurityDescriptor = pSD;
265        sa.bInheritHandle       = TRUE;
266
267        while( !bTpccExit )
268        {
269            if ( hPipe && ReadFile(hPipe, &l, 1, &d,
 NULL) )
270            {
271                DisconnectNamedPipe(hPipe);
272                CloseHandle(hPipe);
273                // open delivery named pipe...
274                hPipe = CreateNamedPipe(lpszPipeName,
 FILE_FLAG_OVERLAPPED | PIPE_ACCESS_DUPLEX,
275                    PIPE_TYPE_BYTE | PIPE_READMODE_BYTE
 | PIPE_NOWAIT,
276                     1, 65535, 65535, 250, &sa);
277            }
278            Sleep( 2000 );  //check for delivery appli-
 cation exit once every 2 seconds.
279        }
280
281        free(pSD);
282
283        return;
284    }
285
286    /* FUNCTION: BOOL WINAPI GetExtensionVer-
 sion(HSE_VERSION_INFO *pVer)
287     *
288     * PURPOSE: This function is called by the inet ser-
 vice when the DLL is first loaded.
289     *
290     * ARGUMENTS:   HSE_VERSION_INFO    *pVer   passed
 in structure in which to place expected version number.
291     *
292     * RETURNS:     TRUE    inet service expected return
 value.
293     *
294     * COMMENTS:    None
295     *
296     */
297
298    BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO
 *pVer)
299    {
300
301        pVer->dwExtensionVersion = MAKEL-
 ONG(HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
302        lstrcpyn(pVer->lpszExtensionDesc, "TPC-C
 Server.", HSE_MAX_EXT_DLL_NAME_LEN);
303
304        return TRUE;
305    }
306
307    /* FUNCTION: DWORD WINAPI HttpExtension-
 Proc(EXTENSION_CONTROL_BLOCK *pECB)
308     *
309     * PURPOSE: This function is the main entry point
 for the TPCC DLL. The internet service
310     *          calls this function passing in the http
 string.
311     *
312     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
313     *                                              service
 information.
314     *
315     * RETURNS:     DWORD
 HSE_STATUS_SUCCESS                  connection can be
 dropped if error
316     *
 HSE_STATUS_SUCCESS_AND_KEEP_CONN    keep connect valid
 comment sent
317     *
318     * COMMENTS:    None
319     *
320     */
321
322    static int
323    ExceptionFilter(int iException, int iLine, char
 *pFile)
324    {
325        Log("thread %d caught exception 0x%x from line
 %d in file %s\n",
326            GetCurrentThreadId(), iException, iLine,
 pFile);
327
328        return EXCEPTION_EXECUTE_HANDLER;
329    }
330
331    DWORD WINAPI HttpExtension-
 Proc(EXTENSION_CONTROL_BLOCK *pECB)
332    {
333        int iCmd, FormId, TermId, iSyncId;
334        FILE *fp;
335    #ifdef TRANS_SEQ
336        int seq;
337    #endif
338
339        if ( iMaxConnections == -1 )
340        {
341            ErrorMessage(pECB,
 ERR_CAN_NOT_SET_MAX_CONNECTIONS, ERR_TYPE_WEBDLL, NULL,
 -1, -1);
342            return HSE_STATUS_SUCCESS;
343        }
344
345        //if registry setting is for html logging then
 show http string passed in.
346        if ( bLog )
347        {
348            SYSTEMTIME  systemTime;
349
350            fp = fopen(szTpccLogPath, "ab");
351
352            GetLocalTime(&systemTime);
353
354            fprintf(fp, "* QUERY * %2.2d/%2.2d/%2.2d
 %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
355                systemTime.wYear, systemTime.wMonth,
 systemTime.wDay,
356                systemTime.wHour, systemTime.wMinute,
 systemTime.wSecond,
357                pECB->lpszQueryString);
358            fclose(fp);
```

```
359         }
360
361
362         //process http query
363         if ( !ProcessQueryString(pECB, &iCmd, &FormId,
 &TermId, &iSyncId) )
364         {
365             if ( TermId < 0 )
366                 ErrorMessage(pECB, ERR_INVALID_TERMID,
 ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
367             else
368                 ErrorMessage(pECB,
 ERR_COMMAND_UNDEFINED, ERR_TYPE_WEBDLL, NULL, TermId,
 iSyncId);
369             return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
370         }
371
372         if ( TermId != 0 )
373         {
374             if ( !IsValidTermId(TermId) )
375             {
376                 fprintf(stderr, "Invalid TermID %d\n");
377                 ErrorMessage(pECB, ERR_INVALID_TERMID,
 ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
378                return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
379             }
380
381           //must have a valid syncid here since termid
 is valid
382             if ( iSyncId < 1 || iSyncId != Term.pClient-
 Data[TermId].iSyncId )
383             {
384                 fprintf(stderr, "Bad TermId SyncId pair
 - Termid=%d SyncId=%d Actual SyncID=%d w_id=%d d_id=%d
 TickCount=%u\n",
385                     TermId, iSyncId, Term.pClient-
 Data[TermId].iSyncId, Term.pClientData[TermId].w_id,
 Term.pClientData[TermId].d_id, Term.pClientData[Ter-
 mId].iTickCount);
386                 ErrorMessage(pECB,
 ERR_INVALID_SYNC_CONNECTION, ERR_TYPE_WEBDLL, NULL, Ter-
 mId, iSyncId);
387                 return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
388             }
389
390         }
391
392         //set use time
393         Term.pClientData[TermId].iTickCount = GetTick-
 Count();
394
395         __try
396         {
397     #ifdef TRANS_SEQ
398             {
399                 char *p = strstr(pECB->lpszQueryString,
 "SEQ=");
400
401                 seq = -1;
402
403                 if (p)
404                 {
405                     seq=strtoul(p+4, NULL, 0);
406
407                     Log("B 0x%x\n", seq);
408                 }
409
410             }
411     #endif
412
413             //go execute http: command
414             (*DoCmd[iCmd])(pECB, FormId, TermId, iSyn-
 cId);
415
416     #ifdef TRANS_SEQ
417             if (seq != -1)
418                 Log("E 0x%x\n", seq);
419     #endif
420         }
421         __except(ExceptionFilter(GetExceptionCode(),
 __LINE__, __FILE__))
422         {
423             fprintf(stderr, "caught an exception in
 HttpExtensionProc\n");
424         }
425         //finish up and keep connection
426         return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
427     }
428
429     /* FUNCTION: static BOOL IsValidTermId(int TermId)
430      *
431      * PURPOSE: This function checks to see of the
 passed in terminal id is valid.
432      *
433      * ARGUMENTS:   int             TermId
 client terminal id
434      *
435      * RETURNS:     BOOL    FALSE
 Terminal ID Invalid
436      *                     TRUE
 Terminal ID valid
437      *
438      * COMMENTS:    None
439      *
440      */
441
442     BOOL IsValidTermId(int TermId)
443     {
444         return (BOOL) ( TermId > 0 && TermId <=
 Term.iAvailable && Term.pClientData[TermId].inUse );
445     }
446
447     /* FUNCTION: BOOL ProcessQue-
 ryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd, int
 *pFormId, int *pTermId, int *pSyncId)
448      *
449      * PURPOSE: This function extracts the relevent
 information out of the http command passed in from
450      *          the browser.
451      *
452      * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
453      *                                                ser-
 vice information.
454      *              int             *pCmd
 returned command id
455      *              int             *pFormId
 returned active form client browser is on
456      *              int             *pTermId
 returned client terminal id
457      *
458      * RETURNS:     BOOL    FALSE
 success
459      *                     TRUE
 command passed in is invalid
460      *
461      * COMMENTS:    If this is the initial connection
 i.e. client is at welcome screen then
462      *              there will not be a terminal id or
 current form id if this is the case
463      *             then the pTermid and pFormid return
 values are undefined.
464      */
465
466     BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK
 *pECB, int *pCmd, int *pFormId, int *pTermId, int *pSyn-
 cId)
467     {
468         char *ptr;
469         char szBuffer[25];
470         char szTmp[25];
471         char *dest = szBuffer;
472         int i;
473
```

```
474
475         if ( (ptr = strstr(pECB->lpszQueryString, "FOR-
 MID=")) )
476             *pFormId = *(ptr+7) & 0x0F;
477
478         if ( (ptr = strstr(pECB->lpszQueryString, "TER-
 MID=")) )
479         {
480             *pTermId = atoi((ptr+7));
481            if ( *pTermId == 0 )    //terminal id 0 used
 internally
482                 *pTermId = -1;
483             if ( *pTermId == -2 )   //login screen
484                 *pTermId = 0;
485         }
486         else
487             *pTermId = 0;
488
489         if ( (ptr = strstr(pECB->lpszQueryString, "SYN-
 CID=")) )
490             *pSyncId = atoi((ptr+7));
491         else
492             *pSyncId = 0;
493
494         if ( !(ptr = strstr(pECB->lpszQueryString,
 "CMD=")) )
495         {
496             ptr = szBuffer;
497             if ( !stricmp(szBuffer, "Default") )
498                 strcpy(szBuffer, "CMD=Begin");
499             switch( *pFormId )
500             {
501                 case WELCOME_FORM:
502                     strcpy(szBuffer, "CMD=Submit");
503                     break;
504                 case MAIN_MENU_FORM:
505                     strcpy(szBuffer, "CMD=NewOrder");
506                     break;
507                 case NEW_ORDER_FORM:
508                 case PAYMENT_FORM:
509                 case DELIVERY_FORM:
510                 case ORDER_STATUS_FORM:
511                 case STOCK_LEVEL_FORM:
512                     if ( !(*pTermId) )
513                         return FALSE;
514                     if ( GetKeyValue(pECB->lpszQue-
 ryString, "PI*", szTmp, sizeof(szTmp)) )
515                         strcpy(szBuffer, "CMD=Process");
516                     else
517                     {
518                         strcpy(szBuffer, "CMD=");
519                       strcat(szBuffer, szCmds[*pFormId
 - NEW_ORDER_FORM]);
520                     }
521                     break;
522                 default:
523                     return FALSE;
524             }
525         }
526
527         ptr += 4;
528
529         while( *ptr && *ptr != '&' )
530             *dest++ = *ptr++;
531         *dest = 0;
532
533         for(i=0; szCmds[i][0]; i++)
534         {
535             if ( !strcmp(szCmds[i], szBuffer) )
536             {
537                 *pCmd = i;
538                 return TRUE;
539             }
540         }
541         return FALSE;
542     }
543
544     /* FUNCTION: void NewOrder-
 Form(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
545      *
546      * PURPOSE: This function wraps the functionality
 needed for the TPC-C New Order Form.
547      *
548      * ARGUMENTS:   int                     iFormId
 unused
549      *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
550      *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
551      *                                              ser-
 vice information.
552      *
553      * RETURNS:     None
554      *
555      * COMMENTS:    None
556      *
557      */
558
559     void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB,
 int iFormId, int iTermId, int iSyncId)
560     {
561         WriteZString(pECB, MakeNewOrderForm(iTermId,
 iSyncId, TRUE, FALSE));
562
563         UNUSEDPARAM(iFormId);
564
565         return;
566     }
567
568     /* FUNCTION: void Payment-
 Form(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
569      *
570      * PURPOSE: This function wraps the functionality
 needed for the TPC-C Payment Form.
571      *
572      * ARGUMENTS:   int                     iFormId
 unused
573      *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
574      *              int                   iSyncId
 sync id of calling browser
575      *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
576      *                                              ser-
 vice information.
577      * RETURNS:     None
578      *
579      * COMMENTS:    None
580      *
581      */
582
583     void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
584     {
585         WriteZString(pECB, MakePaymentForm(iTermId,
 iSyncId, TRUE) );
586
587         UNUSEDPARAM(iFormId);
588
589     }
590
591     /* FUNCTION: void Delivery-
 Form(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
592      *
593      * PURPOSE: This function wraps the functionality
 needed for the TPC-C Delivery Form.
594      *
595      * ARGUMENTS:   int                     iFormId
 unused
596      *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
```

```
597     *             int                       iSyncId
 sync id of calling browser
598     *             EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
599     *                                                ser-
 vice information.
600     * RETURNS:     None
601     *
602     * COMMENTS:    None
603     *
604     */
605
606    void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB,
 int iFormId, int iTermId, int iSyncId)
607    {
608        WriteZString(pECB, MakeDeliveryForm(iTermId,
 iSyncId, TRUE, TRUE) );
609
610        UNUSEDPARAM(iFormId);
611    }
612
613    /* FUNCTION: void OrderStatus-
 Form(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
614     *
615     * PURPOSE: This function wraps the functionality
 needed for the TPC-C Order Status Form.
616     *
617     * ARGUMENTS:   int                       iFormId
 unused
618     *             int                     iTermId    id
 of calling browser, i.e. TERMID= from http command line
619     *              int                       iSyncId
 sync id of calling borwser
620     *             EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
621     *                                                ser-
 vice information.
622     * RETURNS:     None
623     *
624     * COMMENTS:    None
625     *
626     */
627
628    void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB,
 int iFormId, int iTermId, int iSyncId)
629    {
630       WriteZString(pECB, MakeOrderStatusForm(iTermId,
 iSyncId, TRUE) );
631
632        UNUSEDPARAM(iFormId);
633
634    }
635
636    /* FUNCTION: void StockLevel-
 Form(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
637     *
638     * PURPOSE: This function wraps the functionality
 needed for the TPC-C Stock Level Form.
639     *
640     * ARGUMENTS:   int                       iFormId
 unused
641     *             int                     iTermId    id
 of calling browser, i.e. TERMID= from http command line
642     *              int                       iSyncId
 sync id of calling browser
643     *             EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
644     *                                                ser-
 vice information.
645     * RETURNS:     None
646     *
647     * COMMENTS:    None
648     *
649     */
650
```

```
651    void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB,
 int iFormId, int iTermId, int iSyncId)
652    {
653        WriteZString(pECB, MakeStockLevelForm(iTermId,
 iSyncId, TRUE) );
654
655        return;
656    }
657
658    /* FUNCTION: void Exitcmd(EXTENSION_CONTROL_BLOCK
 *pECB, int iFormId, int iTermId, int iSyncId)
659     *
660     * PURPOSE: This function removes a terminal id
 from use, the allocated structure however remains
661     *          valid so the next request for a new cli-
 ent will not require a new memory allocation.
662     *
663     * ARGUMENTS:   int                       iFormId
 unused
664     *             int                     iTermId    id
 of calling browser, i.e. TERMID= from http command line
665     *              int                       iSyncId
 sync id of calling browser
666     *             EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
667     *                                                ser-
 vice information.
668     * RETURNS:     None
669     *
670     * COMMENTS:    None
671     *
672     */
673
674    void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
675    {
676        (*Term.Delete)(pECB, iTermId);
677
678        WriteZString(pECB, MakeWelcomeForm() );
679
680        UNUSEDPARAM(iFormId);
681        UNUSEDPARAM(iSyncId);
682
683        return;
684    }
685
686    /* FUNCTION: void SubmitCmd(EXTENSION_CONTROL_BLOCK
 *pECB, int iFormId, int iTermId, int iSyncId)
687     *
688     * PURPOSE: This function allocated a new terminal
 id in the Term structure array.
689     *
690     * ARGUMENTS:   int                       iFormId
 unused
691     *             int                     iTermId    id
 of calling browser, i.e. TERMID= from http command line
692     *              int                       iSyncId
 sync id of calling browser
693     *             EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
694     *                                                ser-
 vice information.
695     * RETURNS:     None
696     *
697     * COMMENTS:    A terminal id can be allocated but
 still be invalid if the requested warehouse number
698     *            is outside the range specified in the
 registry. This then will force the client id
699     *             to be invalid and an error message
 sent to the users browser.
700     */
701
702    void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
703    {
704        int iCurrent;
705
```

```
706        if ( (iCurrent = (*Term.Add)(pECB, pECB-
 >lpszQueryString)) < 0 )
707        {
708            ErrorMessage(pECB,
 ERR_CANNOT_INIT_TERMINAL, ERR_TYPE_WEBDLL, NULL, iCur-
 rent, iSyncId);
709            return;
710        }
711
712        if ( Term.pClientData[iCurrent].w_id > iMax-
 WareHouses || Term.pClientData[iCurrent].w_id < 1 )
713        {
714            ErrorMessage(pECB, ERR_W_ID_INVALID,
 ERR_TYPE_WEBDLL, NULL, iCurrent, iSyncId);
715            (*Term.Delete)(pECB, iCurrent);
716            return;
717        }
718        if ( Term.pClientData[iCurrent].d_id < 1 ||
 Term.pClientData[iCurrent].d_id > 10 )
719        {
720            ErrorMessage(pECB, ERR_D_ID_INVALID,
 ERR_TYPE_WEBDLL, NULL, iCurrent, iSyncId);
721            (*Term.Delete)(pECB, iCurrent);
722            return;
723        }
724
725        WriteZString(pECB, MakeMainMenuForm(iCurrent,
 Term.pClientData[iCurrent].iSyncId) );
726
727        return;
728    }
729
730    /* FUNCTION: void BeginCmd(EXTENSION_CONTROL_BLOCK
 *pECB, int iFormId, int iTermId, int iSyncId)
731     *
732     * PURPOSE: This function is the first command exe-
 cuted. It is executed with the command
733     *          CMD=Begin?Server=xxx from the http com-
 mand line.
734     *
735     * ARGUMENTS:   int                     iFormId
 unused
736     *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
737     *              int                   iSyncId
 sync id of calling browser
738     *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
739     *                                                ser-
 vice information.
740     * RETURNS:     None
741     *
742     * COMMENTS:    SQL server must be specified, how-
 ever the user and password parameters are optional.
743     *              The complete command line is
 CMD=Begin&Server=server&User=sa&Psw=&. The & are used
744     *              to separate parameters which is
 internet browser standard.
745     */
746
747    void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
748    {
749        LPSTR pQueryString;
750
751        pQueryString = pECB->lpszQueryString;
752
753        if ( !GetKeyValue(pQueryString, "Server",
 szServer, sizeof(szServer)) )
754        {
755           ErrorMessage(pECB, ERR_NO_SERVER_SPECIFIED,
 ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
756            return;
757        }
758
759       if ( !GetKeyValue(pQueryString, "User", szUser,
 sizeof(szUser)) )
760            strcpy(szUser, "sa");
761
762       if ( !GetKeyValue(pQueryString, "Psw", szPass-
 word, sizeof(szPassword)) )
763            strcpy(szPassword, "");
764
765       if ( !GetKeyValue(pQueryString, "Db", szData-
 base, sizeof(szDatabase)) )
766            strcpy(szDatabase, "tpcc");
767
768       WriteZString(pECB, MakeWelcomeForm() );
769
770       UNUSEDPARAM(iFormId);
771
772       return;
773    }
774
775    /* FUNCTION: void Process-
 Cmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
776     *
777     * PURPOSE: This function process the passed in
 http command
778     *
779     * ARGUMENTS:   int                     iFormId
 unused
780     *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
781     *              int                   iSyncId
 sync id of calling browser
782     *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
783     *                                                ser-
 vice information.
784     * RETURNS:     None
785     *
786     * COMMENTS:    None
787     *
788     */
789
790
791    void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
792    {
793        __try
794        {
795        switch( iFormId )
796        {
797            case WELCOME_FORM:
798                return;
799            case MAIN_MENU_FORM:
800                return;
801            case NEW_ORDER_FORM:
802               ProcessNewOrderForm(pECB, iTermId, iSyn-
 cId);
803                return;
804            case PAYMENT_FORM:
805                ProcessPaymentForm(pECB, iTermId, iSyn-
 cId);
806                return;
807            case DELIVERY_FORM:
808               ProcessDeliveryForm(pECB, iTermId, iSyn-
 cId);
809                return;
810            case ORDER_STATUS_FORM:
811                ProcessOrderStatusForm(pECB, iTermId,
 iSyncId);
812                return;
813            case STOCK_LEVEL_FORM:
814                ProcessStockLevelForm(pECB, iTermId,
 iSyncId);
815                return;
816        }
817        }
818        __except(ExceptionFilter(GetExceptionCode(),
 __LINE__, __FILE__))
819        {
```

```
820           fprintf(stderr, "caught an exception in Pro-
 cessCmd\n");
821         }
822
823     }
824
825     /* FUNCTION: void ClearCmd(EXTENSION_CONTROL_BLOCK
 *pECB, int iFormId, int iTermId, int iSyncId)
826      *
827      * PURPOSE: This function frees all currently
 logged in terminal ids.
828      *
829      * ARGUMENTS:   int                    iFormId
 unused
830      *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
831      *              int                   iSyncId
 sync id of calling browser
832      *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
833      *                                                ser-
 vice information.
834      * RETURNS:     None
835      *
836      * COMMENTS:    Use this function with caution, it
 may cause unpredictable results
837      *              if existing browsers attempt to use
 the web client with out
838      *              beginning at the login screen for
 each client.
839      */
840
841     void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
842     {
843         int i;
844
845         EnterCriticalSection(&CriticalSection);
846
847         for(i=0; i<Term.iAvailable; i++)
848         {
849             if ( Term.pClientData[i].inUse )
850                 (*Term.Delete)(pECB, i);
851         }
852
853         Term.iNext          = 0;
854         Term.iAvailable     = 0;
855         Term.iMasterSyncId  = 1;
856
857         if ( Term.pClientData )
858             free(Term.pClientData);
859         Term.pClientData    = NULL;
860         Term.bInit          = FALSE;
861
862         (*Term.Init)();
863         if ( !(*Term.Allocate)() )
864         {
865             ErrorMessage(pECB, ERR_MAX_CONNECT_PARAM,
 ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
866             return;
867         }
868         for(i=Term.iNext; i<Term.iAvailable; i++)
869             Term.pClientData[i].inUse = 0;
870         Term.pClientData[0].inUse = 1;
871
872         LeaveCriticalSection(&CriticalSection);
873
874         WriteZString(pECB, MakeWelcomeForm() );
875
876         return;
877     }
878
879     /* FUNCTION: void MenuCmd(EXTENSION_CONTROL_BLOCK
 *pECB, int iFormId, int iTermId, int iSyncId)
880      *
881      * PURPOSE: This function causes an exit to the main
 menu
882      *
883      * ARGUMENTS:   int                    iFormId
 unused
884      *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
885      *              int                   iSyncId
 sync id of calling browser
886      *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
887      *                                                ser-
 vice information.
888      * RETURNS:     None
889      *
890      * COMMENTS:    None
891      *
892      */
893
894     void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int
 iFormId, int iTermId, int iSyncId)
895     {
896         WriteZString(pECB, MakeMainMenuForm(iTermId,
 iSyncId) );
897
898         return;
899     }
900
901     /* FUNCTION: void NumberOfConnection-
 sCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
 iTermId, int iSyncId)
902      *
903      * PURPOSE: This function returns to the browser
 the total number of active terminal ids
904      *
905      * ARGUMENTS:   int                    iFormId
 unused
906      *             int                   iTermId    id
 of calling browser, i.e. TERMID= from http command line
907      *              int                   iSyncId
 sync id of calling browser
908      *              EXTENSION_CONTROL_BLOCK *pECB
 structure pointer to passed in internet
909      *                                                ser-
 vice information.
910      * RETURNS:     None
911      *
912      * COMMENTS:    None
913      */
914
915     void NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK
 *pECB, int iFormId, int iTermId, int iSyncId)
916     {
917         int i;
918         int iTotal;
919
920         // EnterCriticalSection(&CriticalSection);
921
922         iTotal = 0;
923
924         for(i=0; i<Term.iAvailable; i++)
925         {
926             if ( Term.pClientData[i].inUse )
927                 iTotal++;
928         }
929
930         // LeaveCriticalSection(&CriticalSection);
931
932         h_printf(pECB, "Total Active Connections: %d",
 iTotal);
933
934         return;
935     }
936
937     /* FUNCTION: void
 WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr)
938      *
939      * PURPOSE: This function is the low level output
 function. It writes a string of text back to the
```

```
940     *          client browser.
941     *
942     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 passed in structure pointer from inetsrv.
943     *              char           *szStr         string
 to display in the client browser.
944     *
945     * RETURNS:     None
946     *
947     * COMMENTS:    This function assumes that the
 string to written to the client browser has
948     *              been formatted in an HTML manner.
949     */
950
951    void WriteZString(EXTENSION_CONTROL_BLOCK *pECB,
 char *szStr)
952    {
953        FILE    *fp;
954        int lpbSize;
955        int iSize;
956        char    szHeader[128];
957        char    szHeader1[128];
958
959        lpbSize = strlen(szStr)+1;
960
961        if ( bLog )
962        {
963            SYSTEMTIME  systemTime;
964
965            fp = fopen(szTpccLogPath, "ab");
966
967            GetLocalTime(&systemTime);
968
969           fprintf(fp, "* HTML PAGE * %2.2d/%2.2d/%2.2d
 %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
970                systemTime.wYear, systemTime.wMonth,
 systemTime.wDay,
971                systemTime.wHour, systemTime.wMinute,
 systemTime.wSecond,
972                szStr);
973
974            fclose(fp);
975        }
976
977        iSize = sprintf(szHeader, "200 Ok");
978        sprintf(szHeader1, "Connection: keep-
 alive\r\nContent-type: text/html\r\nContent-length:
 %d\r\n\r\n", lpbSize);
979
980        (*pECB->ServerSupportFunction)(pECB->ConnID,
 HSE_REQ_SEND_RESPONSE_HEADER, szHeader, &iSize, (LPD-
 WORD)szHeader1);
981        (*pECB->WriteClient)(pECB->ConnID, szStr, &lpb-
 Size, 0);
982
983
984        return;
985    }
986
987    /* FUNCTION: void h_printf(EXTENSION_CONTROL_BLOCK
 *pECB, char *format, ...)
988     *
989     * PURPOSE: This function forms a high level printf
 for an HTML browser
990     *
991     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
 passed in structure pointer from inetsrv.
992     *              char           *format        printf
 style format string
993     *               ...                            other
 arguments as required by printf style format string.
994     *
995     * RETURNS:     None
996     *
997     * COMMENTS:    This function is mainly used for
 developmental support.
998     */
 999
1000    static void h_printf(EXTENSION_CONTROL_BLOCK *pECB,
  char *format, ...)
1001    {
1002        char szBuff[512];
1003        char szTmp[512];
1004
1005        va_list marker;
1006        va_start( marker, format );
1007        vsprintf(szTmp, format, marker);
1008        va_end( marker );
1009
1010        wsprintf(szBuff, "<html>%s</html>", szTmp) + 1;
1011
1012        WriteZString(pECB, szBuff);
1013
1014        return;
1015    }
1016
1017    /* FUNCTION: BOOL GetKeyValue(char *pQueryString,
  char *pKey, char *pValue, int iMax)
1018     *
1019     * PURPOSE: This function parses a http formatted
  string for specific key values.
1020     *
1021     * ARGUMENTS:   char           *pQueryString   http
  string from client browser
1022     *              char            *pKey           key
  value to look for
1023     *              char            *pValue         char-
  acter array into which to place key's value
1024     *            int           iMax           maximum
  length of key value array.
1025     *
1026     * RETURNS:     BOOL    FALSE   key value not found
1027     *                      TRUE    key valud found
1028     *
1029     *
1030     * COMMENTS:    http keys are formatted either
  KEY=value& or KEY=value\0. This DLL formats
1031     *             TPC-C input fields in such a manner
  that the keys can be extracted in the
1032     *              above manner.
1033     */
1034
1035    static BOOL GetKeyValue(char *pQueryString, char
  *pKey, char *pValue, int iMax)
1036    {
1037        char *ptr;
1038
1039
1040        if ( !(ptr=strstr(pQueryString, pKey)) )
1041            return FALSE;
1042        if ( !(ptr=strchr(ptr, '=')) )
1043            return FALSE;
1044        ptr++;
1045        iMax--;
1046        while( *ptr && *ptr != '&' && iMax)
1047        {
1048            *pValue++ = *ptr++;
1049            iMax--;
1050        }
1051        *pValue = 0;
1052        return TRUE;
1053    }
1054
1055    /* FUNCTION: void TermInit(void)
1056     *
1057     * PURPOSE: This function initializes the client
  ternimal structure it is called when the TPCC.DLL
1058     *          is first loaded by the inet service.
1059     *
1060     * ARGUMENTS:   none
1061     *
1062     * RETURNS:     None
1063     *
1064     * COMMENTS:    None
```

```
1065     *
1066     */
1067
1068    static void TermInit(void)
1069    {
1070        if ( Term.bInit )
1071            return;
1072        Term.iNext          = 0;
1073        Term.iMasterSyncId  = 1;
1074
1075        Term.iAvailable     = 0;
1076        Term.pClientData    = NULL;
1077        Term.bInit          = TRUE;
1078
1079        return;
1080    }
1081    /* FUNCTION: void TermRestore(void)
1082     *
1083     * PURPOSE: This function frees allocated resources
  associated with the terminal structure.
1084     *
1085     * ARGUMENTS:   none
1086     *
1087     * RETURNS:     None
1088     *
1089     * COMMENTS:    This function is called only with
  the inet service unloads the TPCC.DLL
1090     *
1091     */
1092
1093    static void TermRestore(void)
1094    {
1095        Term.iNext          = 0;
1096        Term.iAvailable     = 0;
1097        Term.iMasterSyncId  = 0;
1098        if ( Term.pClientData )
1099            free(Term.pClientData);
1100        Term.pClientData    = NULL;
1101        Term.bInit          = FALSE;
1102
1103
1104        return;
1105    }
1106
1107    /* FUNCTION: int TermAllocate(void)
1108     *
1109     * PURPOSE: This function allocates more terminal
  array entries in the Term structure.
1110     *
1111     * ARGUMENTS:   None
1112     *
1113     * RETURNS:     int TRUE or 1 if sucessfull
1114     *              int FALSE or 0 if terminal id cannot
  be allocated.
1115     *
1116     * COMMENTS:    None
1117     *
1118     */
1119
1120    static int TermAllocate(void)
1121    {
1122        Term.iAvailable =  iMaxConnections;
1123
1124        if ( Term.pClientData )
1125        {
1126            fprintf(stderr, "trying to realloc in Ter-
  mAllocate()\n");
1127            return 0;
1128        }
1129
1130        Term.pClientData = (PCLIENTDATA)mal-
  loc(Term.iAvailable * sizeof(CLIENTDATA));
1131        return ( Term.pClientData ) ? 1 : 0;
1132    }
1133
1134    /* FUNCTION: int TermAdd(EXTENSION_CONTROL_BLOCK
  *pECB, char *pQueryString)
1135     *
1136     * PURPOSE: This function assigns a terminal id
  which is used to identify a client browser.
1137     *
1138     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK
  *pECB           passed in structure pointer from inetsrv.
1139     *            char                 *pQueryString
  http query string passed to this DLL.
1140     *
1141     * RETURNS:     int     assigned terminal id
1142     *                  -1  cannot assign id error
  occured.
1143     *
1144     *
1145     * COMMENTS:    if the terminal id cannot be
  assigned it is because of insufficient memory or the
1146     *              SQL connection cannot be allocated.
1147     *
1148     */
1149
1150    static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB,
  char *pQueryString)
1151    {
1152        char    szTmp[32];
1153        int     i, iCurrent;
1154        static int iHint=0;
1155
1156        EnterCriticalSection(&CriticalSection);
1157
1158        if (iHint < Term.iAvailable && !Term.pClient-
  Data[iHint].inUse)
1159            iCurrent = iHint;
1160        else
1161        {
1162            int iTotalConnections, iTickCount;
1163            for(i=0, iTotalConnections = 0;
  i<Term.iAvailable; i++)
1164            {
1165                if ( Term.pClientData[i].inUse )
1166                    iTotalConnections++;
1167            }
1168
1169            if ( iTotalConnections >= iMaxConnections )
1170            {
1171                for(iCurrent = 1, i=1, iTickCount =
  0x7FFFFFFF; i<iMaxConnections; i++)
1172                {
1173                    if ( iTickCount > Term.pClient-
  Data[i].iTickCount )
1174                    {
1175                        iTickCount = Term.pClient-
  Data[i].iTickCount;
1176                        iCurrent = i;
1177                    }
1178                }
1179                fprintf(stderr, "iTotalConnections(%d)
  >= iMaxConnections(%d), stealing entry %d\n",
1180                    iTotalConnections, iMaxConnections,
  iCurrent);
1181            }
1182            else
1183            {
1184                for(i=0; i<Term.iAvailable; i++)
1185                {
1186                    if ( !Term.pClientData[i].inUse )
1187                        break;
1188                }
1189                iCurrent = i;
1190            }
1191        }
1192
1193        if ( i == Term.iAvailable )
1194        {
1195            Term.iNext = Term.iAvailable;
1196            if ( !(*Term.Allocate)() )
1197                goto TermAddErr2;
1198            for(i=Term.iNext; i<Term.iAvailable; i++)
```

```
1199                Term.pClientData[i].inUse = 0;
1200            iCurrent = Term.iNext;
1201        }
1202
1203        Term.pClientData[iCurrent].inUse = 1;
1204
1205        if ( !GetKeyValue(pQueryString, "w_id", szTmp,
  sizeof(szTmp)) )
1206            goto TermAddErr1;
1207
1208        Term.pClientData[iCurrent].w_id =
  (short)atoi(szTmp);
1209
1210        if ( !GetKeyValue(pQueryString, "d_id", szTmp,
  sizeof(szTmp)) )
1211            goto TermAddErr1;
1212
1213        Term.pClientData[iCurrent].d_id = atoi(szTmp);
1214
1215        Term.pClientData[iCurrent].iTickCount = Get-
  TickCount();
1216        Term.pClientData[iCurrent].iSyncId = Term.iMas-
  terSyncId++;
1217
1218        if ( Init(pECB, iCurrent, Term.pClient-
  Data[iCurrent].iSyncId, szServer, szUser, szPassword,
  szDatabase) )
1219        {
1220            (*Term.Delete)(pECB, iCurrent);
1221            goto TermAddErr1;
1222        }
1223
1224        if ((iCurrent % 100) == 0 ||
1225            Term.pClientData[iCurrent].iSyncId != iCur-
  rent)
1226            fprintf(stderr, "TermAdd: TermID %d (w_id
  %d, d_id %d) gets Syncid %d, TickCount=%u\n",
1227                iCurrent, Term.pClientData[iCur-
  rent].w_id,
1228                Term.pClientData[iCurrent].d_id,
1229                Term.pClientData[iCurrent].iSyncId,
1230                Term.pClientData[iCurrent].iTickCount);
1231
1232        iHint = iCurrent+1;
1233
1234        LeaveCriticalSection(&CriticalSection);
1235        return iCurrent;
1236
1237    TermAddErr1:
1238        Term.pClientData[iCurrent].inUse = 0;
1239    TermAddErr2:
1240        LeaveCriticalSection(&CriticalSection);
1241        return -1;  //terminal unsuccessfully added
1242    }
1243
1244    /* FUNCTION: void TermDe-
  lete(EXTENSION_CONTROL_BLOCK *pECB, int id)
1245     *
1246     * PURPOSE: This function makes a terminal entry in
  the Term array available for reuse.
1247     *
1248     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
  passed in structure pointer from inetsrv.
1249     *              int                     id      Ter-
  minal id of client exiting
1250     *
1251     * RETURNS:     None
1252     *
1253     * COMMENTS:    None
1254     *
1255     */
1256
1257    static void TermDelete(EXTENSION_CONTROL_BLOCK
  *pECB, int id)
1258    {
1259        if ( id >= 0 && id < Term.iAvailable )
1260        {
1261            Close(pECB, id, -1);
1262            Term.pClientData[id].inUse = 0;
1263        }
1264
1265        return;
1266    }
1267
1268    /* FUNCTION: BOOL Init(EXTENSION_CONTROL_BLOCK
  *pECB, int iTermId, int iSyncId, char *szServer, char
  *szUser, char *szPassword, char *szDatabase)
1269     *
1270     * PURPOSE: This function initializes the sql con-
  nection for use.
1271     *
1272     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
  passed in structure pointer from inetsrv.
1273     *              int                     iTermId     id
  of browser client that this connection is for.
1274     *              int                     iSyncId
  sync id for this client session
1275     *              char                    *szServer
  sql server name
1276     *              char                    *szUser
  user name
1277     *              char                    *szPassword
  user password
1278     *              char                    *szDatabase
  database to use
1279     *
1280     * RETURNS:     BOOL    FALSE   if successfull
1281     *                      TRUE    if an error occurs
  and connection cannot be established.
1282     *
1283     * COMMENTS:    None
1284     *
1285     */
1286
1287    BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTer-
  mId, int iSyncId, char *szServer, char *szUser, char
  *szPassword, char *szDatabase)
1288    {
1289        char    szApp[32];
1290        char    server[256];
1291        char    database[256];
1292        char    user[256];
1293        char    password[256];
1294
1295        sprintf(szApp, "TPCC:%ld", (int)iTermId);
1296
1297        Term.pClientData[iTermId].dbproc = NULL;
1298
1299        sprintf(szApp, "TPCC:%ld", (int)iTermId);
1300
1301        Term.pClientData[iTermId].dbproc = NULL;
1302
1303        strcpy(server, szServer);
1304        strcpy(database, szDatabase);
1305        strcpy(user, szUser);
1306        strcpy(password, szPassword);
1307
1308        if ( SQLOpenConnection(pECB, iTermId, iSyncId,
  &Term.pClientData[iTermId].dbproc, server, database,
  user, password, szApp, &Term.pClientData[iTermId].spid)
  )
1309        {
1310            ErrorMessage(pECB, ERR_SQL_OPEN_CONNECTION,
  ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
1311             return TRUE;
1312        }
1313        return FALSE;
1314    }
1315
1316    /* FUNCTION: BOOL Close(EXTENSION_CONTROL_BLOCK
  *pECB, int iTermId, int iSyncId)
1317     *
1318     * PURPOSE: This function closes the sql connection
  for use.
```

```
1319     *
1320     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
  passed in structure pointer from inetsrv.
1321     *              int                     iTermId id
  of browser client that this connection is for.
1322     *              int                   iSyncId sync
  id of client browser
1323     *
1324     * RETURNS:     BOOL    FALSE   if successfull
1325     *                      TRUE    if an error occurs
  and connection cannot be terminated.
1326     *
1327     * COMMENTS:    None
1328     *
1329     */
1330
1331    static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB,
  int iTermId, int iSyncId)
1332    {
1333        PECBINFO    pEcbInfo;
1334
1335        if (Term.pClientData[iTermId].dbproc != NULL)
1336        {
1337            if ( (pEcbInfo = SQLGetECB(Term.pClient-
  Data[iTermId].dbproc)))
1338            {
1339                pEcbInfo->iTermId = -1;
1340                pEcbInfo->iSyncId = -1;
1341                free(pEcbInfo); //free up user info
1342            }
1343            return SQLCloseConnection(pECB, Term.pCli-
  entData[iTermId].dbproc);
1344        }
1345
1346        UNUSEDPARAM(iSyncId);
1347    }
1348
1349    /* FUNCTION: void FormatString(char *szDest, char
  *szPic, char *szSrc)
1350     *
1351     * PURPOSE: This function formats a character
  string for inclusion in the
1352     *          HTML formatted page being constructed.
1353     *
1354     * ARGUMENTS:   char        *szDest Destination
  buffer where formatted string is to be placed
1355     *              char        *szPic  picture string
  which describes how character value is to be
1356     *                                  formatted.
1357     *              char       *szSrc  character string
  value.
1358     *
1359     * RETURNS:     None
1360     *
1361     * COMMENTS:    This functions is used to format
  TPC-C phone and zip value strings.
1362     *
1363     */
1364
1365    static void FormatString(char *szDest, char *szPic,
  char *szSrc)
1366    {
1367        while( *szPic )
1368        {
1369            if ( *szPic == 'X' )
1370            {
1371                if ( *szSrc )
1372                    *szDest++ = *szSrc++;
1373                else
1374                    *szDest++ = ' ';
1375            }
1376            else
1377                *szDest++ = *szPic;
1378            szPic++;
1379        }
1380        *szDest = 0;
1381
1382        return;
1383    }
1384
1385    /* FUNCTION: char *MakeStockLevelForm(int iTermId,
  int iSyncId, BOOL bInput)
1386     *
1387     * PURPOSE: This function constructs the Stock
  Level HTML page.
1388     *
1389     * ARGUMENTS:   int             iTermId client
  browser terminal id
1390     *            int           iSyncId client browser
  sync id
1391     *             BOOL           bInput  TRUE if form
  is being constructed for input else FALSE
1392     *
1393     * RETURNS:     char *          A pointer to buffer
  inside client structure where HTML form is built.
1394     *
1395     * COMMENTS:    The internal client buffer is cre-
  ated when the terminal id is assigned and should not
1396     *             be freed except when the client ter-
  minal id is no longer needed.
1397     */
1398
1399    static char *MakeStockLevelForm(int iTermId, int
  iSyncId, BOOL bInput)
1400    {
1401        char    *szForm;
1402
1403        szForm = (char *)Term.pClientData[iTer-
  mId].szBuffer;
1404
1405        Term.pClientData[iTermId].stockLevel-
  Data.w_id           = (short)Term.pClientData[iTer-
  mId].w_id;
1406        Term.pClientData[iTermId].stockLevel-
  Data.d_id           = (short)Term.pClientData[iTer-
  mId].d_id;
1407        Term.pClientData[iTermId].stockLevel-
  Data.num_deadlocks  = 0;
1408
1409        strcpy(szForm, "<HTML><HEAD><TITLE>TPC-C Stock
  Level</TITLE></HEAD>");
1410        strcat(szForm, "<FORM ACTION=\"tpcc.dll\"
  METHOD=\"GET\">");
1411        if ( bInput )
1412            strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"PI*\" VALUE=\"\">");
1413        strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"STATUSID\" VALUE=\"0\">");
1414        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
  STOCK_LEVEL_FORM);
1415        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
  mId);
1416        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
  cId);
1417        strcat(szForm,
  "<PRE>                                  Stock-Level<BR>");
1418        wsprintf(szForm+strlen(szForm), "Warehouse:
  %4.4d   District: %2.2d<BR><BR>", Term.pClientData[iTer-
  mId].stockLevelData.w_id, Term.pClientData[iTer-
  mId].stockLevelData.d_id);
1419        if ( bInput )
1420        {
1421            strcat(szForm,  "Stock Level Threshold:
  <INPUT NAME=\"TT*\" SIZE=2><BR><BR>"
1422                                "low stock:    <BR><HR>"
1423                                "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"Process\">"
1424                                "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"Menu\">" );
1425        }
1426        else
```

```
1427        {
1428           wsprintf(szForm+strlen(szForm), "Stock Level
  Threshold: %2.2d<BR><BR>", Term.pClientData[iTer-
  mId].stockLevelData.thresh_hold);
1429
1430           wsprintf(szForm+strlen(szForm), "low stock:
  %3.3d</PRE><BR><HR>", Term.pClientData[iTermId].stock-
  LevelData.low_stock);
1431           strcat(szForm,  "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..NewOrder..\">"
1432                          "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Payment..\">"
1433                          "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Delivery..\">"
1434                          "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Order-Status..\">"
1435                          "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
1436                          "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Exit..\">" );
1437        }
1438
1439        strcat(szForm, "</FORM></HTML>");
1440
1441        return szForm;
1442    }
1443
1444    /* FUNCTION: char *MakeMainMenuForm(int iTermId,
  int iSyncId)
1445     *
1446     * PURPOSE: This function
1447     *
1448     * ARGUMENTS:   int             iTermId client
  browser terminal id
1449     *             int            iSyncId client browser
  sync id
1450     *
1451     * RETURNS:     char *          A pointer to buffer
  inside client structure where HTML form is built.
1452     *
1453     * COMMENTS:    The internal client buffer is cre-
  ated when the terminal id is assigned and should not
1454     *              be freed except when the client ter-
  minal id is no longer needed.
1455     */
1456
1457    static char *MakeMainMenuForm(int iTermId, int
  iSyncId)
1458    {
1459        char    *szForm;
1460
1461        szForm = (char *)Term.pClientData[iTer-
  mId].szBuffer;
1462
1463        strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C Main
  Menu</TITLE></HEAD><BODY>"
1464                        "Select Desired Transac-
  tion.<BR><HR>"
1465                        "<FORM ACTION=\"tpcc.dll\"
  METHOD=\"GET\">");
1466        strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"STATUSID\" VALUE=\"0\">");
1467        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
  mId);
1468        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
  cId);
1469        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
  MAIN_MENU_FORM);
1470        strcat(szForm,  "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..NewOrder..\">"
1471                        "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Payment..\">"
1472                        "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Delivery..\">"
1473                        "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Order-Status..\">"
1474                        "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
1475                        "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Exit..\">"
1476                        "</FORM>"
1477                        "</HTML>" );
1478
1479        return szForm;
1480    }
1481
1482    /* FUNCTION: char *MakeWelcomeForm(void)
1483     *
1484     * PURPOSE: This function
1485     *
1486     * ARGUMENTS:   None
1487     *
1488     * RETURNS:     char *          A pointer to the
  static HTML welcome form.
1489     *
1490     * COMMENTS:    The welcome form is static.
1491     */
1492
1493    static char *MakeWelcomeForm(void)
1494    {
1495        return szWelcomeForm;
1496    }
1497
1498    /* FUNCTION: char *MakeNewOrderForm(int iTermId,
  BOOL bInput, BOOL bValid)
1499     *
1500     * PURPOSE: This function
1501     *
1502     * ARGUMENTS:   int             iTermId client
  browser terminal id
1503     *             int            iSyncId client browser
  sync id
1504     *              BOOL           bInput  TRUE if form
  is being constructed for input else FALSE
1505     *             BOOL          bValid  TRUE if Newor-
  derData valid, ELSE FALSE effects output only
1506     *
1507     * RETURNS:     char *          A pointer to buffer
  inside client structure where HTML form is built.
1508     *
1509     * COMMENTS:    The internal client buffer is cre-
  ated when the terminal id is assigned and should not
1510     *              be freed except when the client ter-
  minal id is no longer needed.
1511     */
1512
1513    static char *MakeNewOrderForm(int iTermId, int
  iSyncId, BOOL bInput, BOOL bValid)
1514    {
1515        char    *szForm;
1516        char    szName[146];
1517        char    szCredit[14];
1518        int     i;
1519
1520        szForm = (char *)Term.pClientData[iTer-
  mId].szBuffer;
1521
1522        Term.pClientData[iTermId].newOrderData.w_id =
  Term.pClientData[iTermId].w_id;
1523
1524        strcpy(szForm,  "<HTML>"
1525                        "<HEAD><TITLE>TPC-C New
  Order</TITLE></HEAD><BODY>"
1526                        "<FORM ACTION=\"tpcc.dll\"
  METHOD=\"GET\">" );
1527
1528        if ( bInput )
1529        {
1530            strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"PI*\" VALUE=\"\">");
1531            strcat(szForm, "<INPUT TYPE=\"hidden\"
```

```
   NAME=\"STATUSID\" VALUE=\"0\">");
1532        }
1533        else
1534        {
1535            if ( bValid )
1536                strcat(szForm, "<INPUT TYPE=\"hidden\"
   NAME=\"STATUSID\" VALUE=\"0\">");
1537            else
1538                wsprintf(szForm+strlen(szForm), "<INPUT
   TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">",
   ERR_BAD_ITEM_ID);
1539        }
1540
1541        wsprintf(szForm+strlen(szForm), "<INPUT
   TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
   NEW_ORDER_FORM);
1542        wsprintf(szForm+strlen(szForm), "<INPUT
   TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
   mId);
1543        wsprintf(szForm+strlen(szForm), "<INPUT
   TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
   cId);
1544        strcat(szForm,
   "<PRE>                                  New Order<BR>");
1545
1546        if ( bInput )
1547        {
1548            wsprintf(szForm+strlen(szForm), "Warehouse:
   %4.4d   District: <INPUT NAME=\"DID*\"
   SIZE=1>                          Date:<BR>", Term.pClient-
   Data[iTermId].newOrderData.w_id);
1549            strcat(szForm,  "Customer:  <INPUT
   NAME=\"CID*\" SIZE=4>   Name:                Credit:
   %Disc:<BR>"
1550                            "Order Number:          Num-
   ber of Lines:          W_tax:         D_tax:<BR><BR>"
1551                    " Supp_W  Item_Id  Item Name
   Qty  Stock  B/G  Price    Amount<BR>"
1552                            " <INPUT NAME=\"SP00*\"
   SIZE=4>  <INPUT NAME=\"IID00*\"
   SIZE=6>                              <INPUT NAME=\"Qty00*\"
   SIZE=1><BR>"
1553                            " <INPUT NAME=\"SP01*\"
   SIZE=4>  <INPUT NAME=\"IID01*\"
   SIZE=6>                              <INPUT NAME=\"Qty01*\"
   SIZE=1><BR>"
1554                            " <INPUT NAME=\"SP02*\"
   SIZE=4>  <INPUT NAME=\"IID02*\"
   SIZE=6>                              <INPUT NAME=\"Qty02*\"
   SIZE=1><BR>"
1555                            " <INPUT NAME=\"SP03*\"
   SIZE=4>  <INPUT NAME=\"IID03*\"
   SIZE=6>                              <INPUT NAME=\"Qty03*\"
   SIZE=1><BR>"
1556                            " <INPUT NAME=\"SP04*\"
   SIZE=4>  <INPUT NAME=\"IID04*\"
   SIZE=6>                              <INPUT NAME=\"Qty04*\"
   SIZE=1><BR>"
1557                            " <INPUT NAME=\"SP05*\"
   SIZE=4>  <INPUT NAME=\"IID05*\"
   SIZE=6>                              <INPUT NAME=\"Qty05*\"
   SIZE=1><BR>"
1558                            " <INPUT NAME=\"SP06*\"
   SIZE=4>  <INPUT NAME=\"IID06*\"
   SIZE=6>                              <INPUT NAME=\"Qty06*\"
   SIZE=1><BR>"
1559                            " <INPUT NAME=\"SP07*\"
   SIZE=4>  <INPUT NAME=\"IID07*\"
   SIZE=6>                              <INPUT NAME=\"Qty07*\"
   SIZE=1><BR>"
1560                            " <INPUT NAME=\"SP08*\"
   SIZE=4>  <INPUT NAME=\"IID08*\"
   SIZE=6>                              <INPUT NAME=\"Qty08*\"
   SIZE=1><BR>"
1561                            " <INPUT NAME=\"SP09*\"
   SIZE=4>  <INPUT NAME=\"IID09*\"
   SIZE=6>                              <INPUT NAME=\"Qty09*\"
   SIZE=1><BR>"
1562                            " <INPUT NAME=\"SP10*\"
   SIZE=4>  <INPUT NAME=\"IID10*\"
   SIZE=6>                              <INPUT NAME=\"Qty10*\"
   SIZE=1><BR>"
1563                            " <INPUT NAME=\"SP11*\"
   SIZE=4>  <INPUT NAME=\"IID11*\"
   SIZE=6>                              <INPUT NAME=\"Qty11*\"
   SIZE=1><BR>"
1564                            " <INPUT NAME=\"SP12*\"
   SIZE=4>  <INPUT NAME=\"IID12*\"
   SIZE=6>                              <INPUT NAME=\"Qty12*\"
   SIZE=1><BR>"
1565                            " <INPUT NAME=\"SP13*\"
   SIZE=4>  <INPUT NAME=\"IID13*\"
   SIZE=6>                              <INPUT NAME=\"Qty13*\"
   SIZE=1><BR>"
1566                            " <INPUT NAME=\"SP14*\"
   SIZE=4>  <INPUT NAME=\"IID14*\"
   SIZE=6>                              <INPUT NAME=\"Qty14*\"
   SIZE=1><BR>"
1567                            "Execution Sta-
   tus:
   Total:<BR><HR>"
1568                            "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"Process\">"
1569                            "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"Menu\">"
1570                            "</FORM>"
1571                            "</HTML>" );
1572
1573        }
1574        else
1575        {
1576            if ( bValid )
1577            {
1578                wsprintf(szForm+strlen(szForm), "Ware-
   house: %4.4d   District: %2.2d                    Date:
   %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d <BR>",
1579                    Term.pClientData[iTermId].newOrder-
   Data.w_id,
1580                    Term.pClientData[iTermId].newOrder-
   Data.d_id,
1581                    Term.pClientData[iTermId].newOrder-
   Data.o_entry_d.day,
1582                    Term.pClientData[iTermId].newOrder-
   Data.o_entry_d.month,
1583                    Term.pClientData[iTermId].newOrder-
   Data.o_entry_d.year,
1584                    Term.pClientData[iTermId].newOrder-
   Data.o_entry_d.hour,
1585                    Term.pClientData[iTermId].newOrder-
   Data.o_entry_d.minute,
1586                    Term.pClientData[iTermId].newOrder-
   Data.o_entry_d.second);
1587            }
1588            else
1589            {
1590                wsprintf(szForm+strlen(szForm), "Ware-
   house: %4.4d   District: %2.2d
   Date:<BR>",
1591                    Term.pClientData[iTermId].newOrder-
   Data.w_id,
1592                    Term.pClientData[iTermId].newOrder-
   Data.d_id);
1593            }
1594
1595            FormatHTMLString(szName, Term.pClient-
   Data[iTermId].newOrderData.c_last, 16),
1596            FormatHTMLString(szCredit, Term.pClient-
   Data[iTermId].newOrderData.c_credit, 2);
1597
1598           wsprintf(szForm+strlen(szForm), "Customer:
   %4.4d   Name: %s   Credit: %s   ",
1599                Term.pClientData[iTermId].newOrder-
   Data.c_id, szName, szCredit);
1600
```

```
1601
1602            if ( bValid )
1603            {
1604                sprintf(szForm+strlen(szForm), "%%Disc:
  %5.2f            <BR>", Term.pClientData[iTermId].newOr-
  derData.c_discount);
1605                sprintf(szForm+strlen(szForm), "Order
  Number: %8.8d  Number of Lines: %2.2d      W_tax: %5.2f
  D_tax: %5.2f  <BR><BR>",
1606                    Term.pClientData[iTermId].newOrder-
  Data.o_id,
1607                    Term.pClientData[iTermId].newOrder-
  Data.o_ol_cnt,
1608                    Term.pClientData[iTermId].newOrder-
  Data.w_tax,
1609                    Term.pClientData[iTermId].newOrder-
  Data.d_tax);
1610
1611                strcat(szForm, " Supp_W  Item_Id  Item
  Name                Qty  Stock  B/G  Price
  Amount<BR>");
1612                for(i=0; i<Term.pClientData[iTer-
  mId].newOrderData.o_ol_cnt; i++)
1613                {
1614                    FormatHTMLString(szName, Term.pCli-
  entData[iTermId].newOrderData.Ol[i].ol_i_name, 24);
1615
1616                    sprintf(szForm+strlen(szForm), "
  %4.4d   %6.6d   %s  %2.2d    %3.3d    %1.1s   $%6.2f
  $%7.2f  <BR>",
1617                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_supply_w_id,
1618                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_i_id,
1619                         szName,
1620                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_quantity,
1621                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_stock,
1622                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_brand_generic,
1623                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_i_price,
1624                        Term.pClientData[iTermId].newOr-
  derData.Ol[i].ol_amount );
1625                }
1626            }
1627            else
1628            {
1629                strcat(szForm, "%Disc:<BR>");
1630                sprintf(szForm+strlen(szForm), "Order
  Number: %8.8d  Number of Lines:
  W_tax:         D_tax:<BR><BR>",
1631                    Term.pClientData[iTermId].newOrder-
  Data.o_id);
1632
1633                strcat(szForm, " Supp_W  Item_Id  Item
  Name                Qty  Stock  B/G  Price
  Amount<BR>");
1634
1635                i = 0;
1636            }
1637            for(; i<15; i++)
1638                strcat(szForm, "<BR>");
1639
1640            if ( bValid )
1641            {
1642                sprintf(szForm+strlen(szForm), "Execu-
  tion Status: %24.24s                  Total:  $%8.2f  ",
1643                    Term.pClientData[iTermId].newOrder-
  Data.execution_status,
1644                    Term.pClientData[iTermId].newOrder-
  Data.total_amount);
1645            }
1646            else
1647            {
1648                sprintf(szForm+strlen(szForm), "Execu-
  tion Status: %24.24s                  Total:",
1649                    Term.pClientData[iTermId].newOrder-
  Data.execution_status);
1650            }
1651
1652            strcat(szForm,  "</PRE><HR><BR>"
1653                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..NewOrder..\">"
1654                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Payment..\">"
1655                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Delivery..\">"
1656                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Order-Status..\">"
1657                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
1658                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Exit..\">" );
1659            strcat(szForm, "</FORM></HTML>");
1660        }
1661
1662        return szForm;
1663
1664    }
1665
1666    /* FUNCTION: char *MakePaymentForm(int iTermId, int
  iSyncId, BOOL bInput)
1667     *
1668     * PURPOSE: This function
1669     *
1670     * ARGUMENTS:   int             iTermId client
  browser terminal id
1671     *            int           iSyncId client browser
  sync id
1672     *             BOOL           bInput  TRUE if form
  is being constructed for input else FALSE
1673     *
1674     * RETURNS:     char *          A pointer to buffer
  inside client structure where HTML form is built.
1675     *
1676     * COMMENTS:    The internal client buffer is cre-
  ated when the terminal id is assigned and should not
1677     *             be freed except when the client ter-
  minal id is no longer needed.
1678     */
1679
1680    static char *MakePaymentForm(int iTermId, int iSyn-
  cId, BOOL bInput)
1681    {
1682        char    *szForm;
1683        char    *ptr;
1684        char    szTmp[64];
1685        char    szW_Zip[26];
1686        char    szD_Zip[26];
1687        char    szC_Zip[26];
1688        char    szC_Phone[26];
1689        char    szTmpStr1[122];
1690        char    szTmpStr2[122];
1691        char    szTmpStr3[122];
1692        char    szTmpStr4[122];
1693        int     i;
1694        int     l;
1695        char    *szZipPic = "XXXXX-XXXX";
1696
1697        szForm = (char *)Term.pClientData[iTer-
  mId].szBuffer;
1698
1699        Term.pClientData[iTermId].paymentData.w_id =
  Term.pClientData[iTermId].w_id;
1700
1701        strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C Pay-
  ment</TITLE></HEAD><BODY>"
1702                        "<FORM ACTION=\"tpcc.dll\"
  METHOD=\"GET\">");
1703        if ( bInput )
1704            strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"PI*\" VALUE=\"\">");
```

```
1705
1706        strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"STATUSID\" VALUE=\"0\">");
1707        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
  PAYMENT_FORM);
1708        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
  mId);
1709        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
  cId);
1710
1711        strcat(szForm,
  "<PRE>                                  Payment<BR>");
1712
1713        if ( bInput )
1714            strcat(szForm,  "Date:<BR><BR>" );
1715        else
1716        {
1717            wsprintf(szForm+strlen(szForm), "Date:
  %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d <BR><BR>",
1718                Term.pClientData[iTermId].payment-
  Data.h_date.day,
1719                Term.pClientData[iTermId].payment-
  Data.h_date.month,
1720                Term.pClientData[iTermId].payment-
  Data.h_date.year,
1721                Term.pClientData[iTermId].payment-
  Data.h_date.hour,
1722                Term.pClientData[iTermId].payment-
  Data.h_date.minute,
1723                Term.pClientData[iTermId].payment-
  Data.h_date.second);
1724        }
1725        wsprintf(szForm+strlen(szForm), "Warehouse:
  %4.4d", Term.pClientData[iTermId].paymentData.w_id);
1726
1727        if ( bInput )
1728        {
1729            strcat(szForm,  "
  District: <INPUT NAME=\"DID*\"
  SIZE=1><BR><BR><BR><BR><BR>"
1730                            "Customer: <INPUT
  NAME=\"CID*\" SIZE=4>"
1731                            "Cust-Warehouse: <INPUT
  NAME=\"CWI*\" SIZE=4>  "
1732                            "Cust-District: <INPUT
  NAME=\"CDI*\" SIZE=1><BR>"
1733                          "Name:
  <INPUT NAME=\"CLT*\" SIZE=16>
  Since:<BR>"
1734
  "
  Credit:<BR>"
1735
  "                                              Disc:<BR>"
1736
  "
  Phone:<BR><BR>"
1737                      "Amount Paid:         $<INPUT
  NAME=\"HAM*\" SIZE=7>      New Cust Balance:<BR>"
1738                        "Credit Limit:<BR><BR>Cust-
  Data: <BR><BR><BR><BR></PRE><HR>"
1739                        "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"Process\"><INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"Menu\">"
1740                        "</BODY></FORM></HTML>" );
1741        }
1742        else
1743        {
1744            sprintf(szForm+strlen(szForm),
  "                       District: %2.2d<BR>",
1745                Term.pClientData[iTermId].payment-
  Data.d_id);
1746
1747            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.w_street_1, 20);
1748            FormatHTMLString(szTmpStr2, Term.pClient-
  Data[iTermId].paymentData.d_street_1, 20);
1749
1750
  sprintf(szForm+strlen(szForm),"%s
  %s<BR>", szTmpStr1, szTmpStr2);
1751
1752            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.w_street_2, 20);
1753            FormatHTMLString(szTmpStr2, Term.pClient-
  Data[iTermId].paymentData.d_street_2, 20);
1754
1755
  sprintf(szForm+strlen(szForm),"%s
  %s<BR>", szTmpStr1, szTmpStr2);
1756
1757            FormatString(szW_Zip, szZipPic, Term.pCli-
  entData[iTermId].paymentData.w_zip);
1758            FormatString(szD_Zip, szZipPic, Term.pCli-
  entData[iTermId].paymentData.d_zip);
1759
1760            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.w_city, 20);
1761            FormatHTMLString(szTmpStr2, Term.pClient-
  Data[iTermId].paymentData.w_state, 2);
1762            FormatHTMLString(szTmpStr3, Term.pClient-
  Data[iTermId].paymentData.d_city, 20);
1763            FormatHTMLString(szTmpStr4, Term.pClient-
  Data[iTermId].paymentData.d_state, 2);
1764
1765            wsprintf(szForm+strlen(szForm), "%s %s
  %10.10s       %s %s %10.10s<BR><BR>",
1766                szTmpStr1, szTmpStr2, szW_Zip,
  szTmpStr3, szTmpStr4, szD_Zip );
1767
1768            wsprintf(szForm+strlen(szForm), "Customer:
  %4.4d  Cust-Warehouse: %4.4d  Cust-District: %2.2d<BR>",
1769                Term.pClientData[iTermId].payment-
  Data.c_id,
1770                Term.pClientData[iTermId].payment-
  Data.c_w_id,
1771                Term.pClientData[iTermId].payment-
  Data.c_d_id);
1772
1773            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.c_first, 16);
1774            FormatHTMLString(szTmpStr2, Term.pClient-
  Data[iTermId].paymentData.c_middle, 2);
1775            FormatHTMLString(szTmpStr3, Term.pClient-
  Data[iTermId].paymentData.c_last, 16);
1776
1777          wsprintf(szForm+strlen(szForm), "Name:   %s
  %s %s     Since:  %2.2d-%2.2d-%4.4d<BR>",
1778                szTmpStr1, szTmpStr2, szTmpStr3,
1779                Term.pClientData[iTermId].payment-
  Data.c_since.day,
1780                Term.pClientData[iTermId].payment-
  Data.c_since.month,
1781                Term.pClientData[iTermId].payment-
  Data.c_since.year);
1782
1783            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.c_street_1, 20);
1784            FormatHTMLString(szTmpStr2, Term.pClient-
  Data[iTermId].paymentData.c_credit, 2);
1785
1786       wsprintf(szForm+strlen(szForm), "    %s
  Credit: %s<BR>", szTmpStr1, szTmpStr2);
1787
1788            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.d_street_2, 20);
1789        sprintf(szForm+strlen(szForm), "    %s
  %%Disc:  %5.2f<BR>",
1790                szTmpStr1, Term.pClientData[iTer-
  mId].paymentData.c_discount);
1791
```

```
1792            FormatString(szC_Zip, szZipPic, Term.pCli-
  entData[iTermId].paymentData.c_zip);
1793            FormatString(szC_Phone, "XXXXXX-XXX-XXX-
  XXXX", Term.pClientData[iTermId].paymentData.c_phone);
1794
1795            FormatHTMLString(szTmpStr1, Term.pClient-
  Data[iTermId].paymentData.c_city, 20);
1796            FormatHTMLString(szTmpStr2, Term.pClient-
  Data[iTermId].paymentData.c_state, 2);
1797
1798           wsprintf(szForm+strlen(szForm), "        %s
  %s %10.10s       Phone:  %-19.19s<BR><BR>",
1799                szTmpStr1, szTmpStr2, szC_Zip, szC_Phone
  );
1800
1801       sprintf(szForm+strlen(szForm), "Amount Paid:
  $%7.2f      New Cust Balance: $%14.2f<BR>",
1802                Term.pClientData[iTermId].payment-
  Data.h_amount,
1803                Term.pClientData[iTermId].payment-
  Data.c_balance);
1804
1805            sprintf(szForm+strlen(szForm), "Credit
  Limit:   $%13.2f<BR><BR>",
1806                Term.pClientData[iTermId].payment-
  Data.c_credit_lim);
1807
1808            ptr = Term.pClientData[iTermId].payment-
  Data.c_credit;
1809            if ( *ptr == 'B' && *(ptr+1) == 'C' )
1810            {
1811               ptr = Term.pClientData[iTermId].payment-
  Data.c_data;
1812                l = strlen( ptr ) / 50;
1813                for(i=0; i<4; i++, ptr += 50)
1814                {
1815                    if ( i <= l )
1816                        UtilStrCpy(szTmp, ptr, 50);
1817                    else
1818                        szTmp[0] = 0;
1819                    if ( !i )
1820                    {
1821                        FormatHTMLString(szTmpStr1,
  szTmp, 50);
1822                       wsprintf(szForm+strlen(szForm),
  "Cust-Data: %s<BR>", szTmpStr1);
1823                    }
1824                    else
1825                    {
1826                        FormatHTMLString(szTmpStr1,
  szTmp, 50);
1827                       wsprintf(szForm+strlen(szForm),
  "           %s<BR>", szTmpStr1);
1828                    }
1829                }
1830            }
1831            else
1832                strcat(szForm, "Cust-Data:
  <BR><BR><BR><BR>");
1833
1834            strcat(szForm,  "</PRE><HR><BR>"
1835                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..NewOrder..\">"
1836                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Payment..\">"
1837                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Delivery..\">"
1838                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Order-Status..\">"
1839                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
1840                           "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Exit..\">"
1841                           "</BODY></FORM></HTML>");
1842        }
1843
1844        return szForm;
1845    }
1846
1847    /* FUNCTION: char *MakeOrderStatusForm(int iTermId,
  int iSyncId, BOOL bInput)
1848     *
1849     * PURPOSE: This function
1850     *
1851     * ARGUMENTS:   int             iTermId client
  browser terminal id
1852     *            int            iSyncId client browser
  sync id
1853     *             BOOL           bInput  TRUE if form
  is being constructed for input else FALSE
1854     *
1855     * RETURNS:     char *          A pointer to buffer
  inside client structure where HTML form is built.
1856     *
1857     * COMMENTS:    The internal client buffer is cre-
  ated when the terminal id is assigned and should not
1858     *             be freed except when the client ter-
  minal id is no longer needed.
1859     */
1860
1861    static char *MakeOrderStatusForm(int iTermId, int
  iSyncId, BOOL bInput)
1862    {
1863        char    *szForm;
1864        char    c_first[98];
1865        char    c_middle[14];
1866        char    c_last[98];
1867        int     i;
1868
1869        szForm = (char *)Term.pClientData[iTer-
  mId].szBuffer;
1870
1871        Term.pClientData[iTermId].orderStatusData.w_id
  = Term.pClientData[iTermId].w_id;
1872
1873        strcpy(szForm,  "<HTML><HEAD><TITLE>TPC-C
  Order-Status</TITLE></HEAD><BODY>"
1874                        "<FORM ACTION=\"tpcc.dll\"
  METHOD=\"GET\">");
1875
1876        if ( bInput )
1877            strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"PI*\" VALUE=\"\">");
1878
1879        strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"STATUSID\" VALUE=\"0\">");
1880        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
  ORDER_STATUS_FORM);
1881        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
  mId);
1882        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
  cId);
1883
1884        strcat(szForm,
  "<PRE>                                Order-Status<BR>" );
1885        wsprintf(szForm+strlen(szForm), "Warehouse:
  %4.4d   ", Term.pClientData[iTermId].orderStatus-
  Data.w_id);
1886
1887        if ( bInput )
1888        {
1889            strcat(szForm,  "District: <INPUT
  NAME=\"DID*\" SIZE=1><BR>"
1890                           "Customer: <INPUT
  NAME=\"CID*\" SIZE=4>   Name:                    <INPUT
  NAME=\"CLT*\" SIZE=23><BR>"
1891                           "Cust-Balance:<BR><BR>"
1892                           "Order-Number:
  Entry-Date:                    Carrier-Number:<BR>"
1893                       "Supply-W    Item-Id    Qty
  Amount      Delivery-Date<BR></PRE>"
```

```
1894                            "<HR><INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"Process\"><INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"Menu\">"
1895                            "</BODY></FORM></HTML>" );
1896        }
1897        else
1898        {
1899            wsprintf(szForm+strlen(szForm), "District:
  %2.2d<BR>", Term.pClientData[iTermId].orderStatus-
  Data.d_id);
1900
1901            FormatHTMLString(c_first, Term.pClient-
  Data[iTermId].orderStatusData.c_first, 16);
1902            FormatHTMLString(c_middle, Term.pClient-
  Data[iTermId].orderStatusData.c_middle, 2);
1903            FormatHTMLString(c_last, Term.pClient-
  Data[iTermId].orderStatusData.c_last, 16);
1904
1905            wsprintf(szForm+strlen(szForm), "Customer:
  %4.4d   Name: %s %s %s<BR>",
1906                Term.pClientData[iTermId].orderStatus-
  Data.c_id, c_first, c_middle, c_last);
1907
1908            sprintf(szForm+strlen(szForm), "Cust-Bal-
  ance: $%9.2f<BR><BR>",
1909                Term.pClientData[iTermId].orderStatus-
  Data.c_balance);
1910
1911            wsprintf(szForm+strlen(szForm), "Order-Num-
  ber: %8.8d   Entry-Date: %2.2d-%2.2d-%4.4d
  %2.2d:%2.2d:%2.2d   Carrier-Number: %2.2d<BR>",
1912                Term.pClientData[iTermId].orderStatus-
  Data.o_id,
1913                Term.pClientData[iTermId].orderStatus-
  Data.o_entry_d.day,
1914                Term.pClientData[iTermId].orderStatus-
  Data.o_entry_d.month,
1915                Term.pClientData[iTermId].orderStatus-
  Data.o_entry_d.year,
1916                Term.pClientData[iTermId].orderStatus-
  Data.o_entry_d.hour,
1917                Term.pClientData[iTermId].orderStatus-
  Data.o_entry_d.minute,
1918                Term.pClientData[iTermId].orderStatus-
  Data.o_entry_d.second,
1919                Term.pClientData[iTermId].orderStatus-
  Data.o_carrier_id);
1920            strcat(szForm+strlen(szForm), "Supply-W
  Item-Id    Qty     Amount      Delivery-Date<BR>");
1921
1922            for(i=0; i<Term.pClientData[iTermId].order-
  StatusData.o_ol_cnt; i++)
1923            {
1924            sprintf(szForm+strlen(szForm), "  %4.4d
  %6.6d     %2.2d     $%8.2f      %2.2d-%2.2d-%4.4d<BR>",
1925                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_supply_w_id,
1926                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_i_id,
1927                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_quantity,
1928                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_amount,
1929                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_delivery_d.day,
1930                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_delivery_d.month,
1931                    Term.pClientData[iTermId].orderSta-
  tusData.OlOrderStatusData[i].ol_delivery_d.year);
1932            }
1933
1934            strcat(szForm,  "<BR></PRE><HR><INPUT
  TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">"
1935                            "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Payment..\">"
1936                            "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Delivery..\">"
1937                            "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Order-Status..\">"
1938                            "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
1939                            "<INPUT TYPE=\"submit\"
  NAME=\"CMD\" VALUE=\"..Exit..\">"
1940                            "</BODY></FORM></HTML>" );
1941        }
1942
1943        return szForm;
1944    }
1945
1946    /* FUNCTION: char *MakeDeliveryForm(int iTermId,
  int iSyncId, BOOL bInput, BOOL bSuccess)
1947     *
1948     * PURPOSE: This function
1949     *
1950     * ARGUMENTS:   int             iTermId client
  browser terminal id
1951     *            int            iSyncId client browser
  sync id
1952     *             BOOL            bInput  TRUE if form
  is being constructed for input else FALSE
1953     *             BOOL            bSuccess TRUE if
  Delivery succeeded else FALSE
1954     *
1955     * RETURNS:     char *          A pointer to buffer
  inside client structure where HTML form is built.
1956     *
1957     * COMMENTS:    The internal client buffer is cre-
  ated when the terminal id is assigned and should not
1958     *             be freed except when the client ter-
  minal id is no longer needed.
1959     */
1960
1961    static char *MakeDeliveryForm(int iTermId, int
  iSyncId, BOOL bInput, BOOL bSuccess)
1962    {
1963        char    *szForm;
1964
1965        szForm = (char *)Term.pClientData[iTer-
  mId].szBuffer;
1966
1967        Term.pClientData[iTermId].deliveryData.w_id =
  Term.pClientData[iTermId].w_id;
1968
1969        strcpy( szForm, "<HTML><HEAD><TITLE>TPC-C
  Delivery</TITLE></HEAD><BODY>"
1970                        "<FORM ACTION=\"tpcc.dll\"
  METHOD=\"GET\">");
1971
1972        if ( bInput )
1973        {
1974            strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"PI*\" VALUE=\"\">");
1975            strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"STATUSID\" VALUE=\"0\">");
1976        }
1977        else
1978        {
1979            if ( !bSuccess )
1980                sprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">",
  ERR_TYPE_DELIVERY_POST);
1981            else
1982                strcat(szForm, "<INPUT TYPE=\"hidden\"
  NAME=\"STATUSID\" VALUE=\"0\">");
1983        }
1984
1985        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">",
  DELIVERY_FORM);
1986        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTer-
  mId);
1987        wsprintf(szForm+strlen(szForm), "<INPUT
  TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyn-
```

```
   cId);
1988
1989        strcat(szForm,
   "<PRE>                                    Delivery<BR>" );
1990
1991        wsprintf(szForm+strlen(szForm), "Warehouse:
   %4.4d<BR><BR>", Term.pClientData[iTermId].delivery-
   Data.w_id);
1992
1993        if ( bInput )
1994            strcat( szForm, "Carrier Number: <INPUT
   NAME=\"OCD*\" SIZE=1><BR><BR>");
1995        else
1996        {
1997            wsprintf(szForm+strlen(szForm), "Carrier
   Number: %2.2d<BR><BR>",
1998                Term.pClientData[iTermId].delivery-
   Data.o_carrier_id);
1999        }
2000        if ( bInput )
2001        {
2002            strcat( szForm, "Execution Sta-
   tus:<BR></PRE>"
2003                                "<HR><INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"Process\">"
2004                                "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"Menu\">" );
2005        }
2006        else
2007        {
2008            wsprintf(szForm+strlen(szForm), "Execution
   Status: %25.25s<BR></PRE>",
2009                Term.pClientData[iTermId].delivery-
   Data.execution_status);
2010
2011           strcat(szForm,  "<HR><INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"..NewOrder..\">"
2012                                "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"..Payment..\">"
2013                                "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"..Delivery..\">"
2014                                "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"..Order-Status..\">"
2015                                "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
2016                                "<INPUT TYPE=\"submit\"
   NAME=\"CMD\" VALUE=\"..Exit..\">" );
2017        }
2018
2019        strcat( szForm, "</BODY></FORM></HTML>" );
2020
2021        return szForm;
2022    }
2023
2024    /* FUNCTION: void ProcessNewOrder-
   Form(EXTENSION_CONTROL_BLOCK   *pECB, int iTermId, int
   iSyncId)
2025     *
2026     * PURPOSE: This function gets and validates the
   input data from the new order form
2027     *         filling in the required input variables.
   it then calls the SQLNewOrder
2028     *          transaction, constructs the output form
   and writes it back to client
2029     *           browser.
2030     *
2031     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
   passed in structure pointer from inetsrv.
2032     *             int                   iTermId client
   browser terminal id
2033     *             int                   iSyncId client
   browser sync id
2034     *
2035     * RETURNS:     None
2036     *
2037     * COMMENTS:    None
2038     *
2039     */
2040
2041    static void ProcessNewOrder-
   Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
   iSyncId)
2042    {
2043        int         iRc;
2044        int         iError;
2045        PECBINFO    pEcbInfo;
2046
2047       memset(&Term.pClientData[iTermId].newOrderData,
   0, sizeof(NEW_ORDER_DATA));
2048
2049        Term.pClientData[iTermId].newOrderData.w_id =
   Term.pClientData[iTermId].w_id;
2050
2051        if ( (iError=GetNewOrderData(pECB->lpszQue-
   ryString, &Term.pClientData[iTermId].newOrderData)) !=
   ERR_SUCCESS )
2052        {
2053           ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL,
   NULL, iTermId, iSyncId);
2054            return;
2055        }
2056
2057        iRc = SQLNewOrder(pECB, iTermId, iSyncId,
   Term.pClientData[iTermId].dbproc, &Term.pClient-
   Data[iTermId].newOrderData, iDeadlockRetry);
2058
2059    #ifdef USE_ODBC
2060        #if (ODBCVER >= 0x0300)
2061            if ( bConnectionPooling && iRc != -3 )
2062                SQLDisconnect(Term.pClientData[iTer-
   mId].dbproc->hdbc);
2063        #endif
2064    #endif
2065
2066        if ((pEcbInfo = SQLGetECB(Term.pClient-
   Data[iTermId].dbproc)) &&
2067            pEcbInfo->bFailed)
2068            return;
2069
2070        if ( iRc < 0 )
2071            ErrorMessage(pECB,
   ERR_NEW_ORDER_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL,
   iTermId, iSyncId);
2072        else
2073            WriteZString(pECB, MakeNewOrderForm(iTer-
   mId, iSyncId, FALSE, (BOOL)iRc) );
2074
2075        return;
2076    }
2077
2078    /* FUNCTION: void ProcessPayment-
   Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
   iSyncId)
2079     *
2080     * PURPOSE: This function gets and validates the
   input data from the payment form
2081     *         filling in the required input variables.
   It then calls the SQLPayment
2082     *          transaction, constructs the output form
   and writes it back to client
2083     *           browser.
2084     *
2085     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
   passed in structure pointer from inetsrv.
2086     *             int                   iTermId client
   browser terminal id
2087     *             int                   iSyncId client
   browser sync id
2088     *
2089     * RETURNS:     None
2090     *
2091     * COMMENTS:    None
2092     *
2093     */
```

```
2094
2095    static void ProcessPayment-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2096    {
2097        int         iRc;
2098        int         iError;
2099        PECBINFO    pEcbInfo;
2100
2101
2102        memset(&Term.pClientData[iTermId].paymentData,
  0, sizeof(PAYMENT_DATA));
2103
2104        Term.pClientData[iTermId].paymentData.w_id =
  Term.pClientData[iTermId].w_id;
2105
2106        if ( (iError=GetPaymentData(pECB->lpszQue-
  ryString, &Term.pClientData[iTermId].paymentData)) !=
  ERR_SUCCESS )
2107        {
2108            ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL,
  NULL, iTermId, iSyncId);
2109            return;
2110        }
2111
2112        iRc = SQLPayment(pECB, iTermId, iSyncId,
  Term.pClientData[iTermId].dbproc, &Term.pClient-
  Data[iTermId].paymentData, iDeadlockRetry);
2113
2114    #ifdef USE_ODBC
2115        #if (ODBCVER >= 0x0300)
2116            if ( bConnectionPooling && iRc != -3 )
2117                SQLDisconnect(Term.pClientData[iTer-
  mId].dbproc->hdbc);
2118        #endif
2119    #endif
2120
2121        if ((pEcbInfo = SQLGetECB(Term.pClient-
  Data[iTermId].dbproc)) &&
2122            pEcbInfo->bFailed)
2123            return;
2124
2125        if ( iRc == 0 )
2126            ErrorMessage(pECB,
  ERR_PAYMENT_INVALID_CUSTOMER, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2127        else if ( iRc < 0 )
2128            ErrorMessage(pECB,
  ERR_PAYMENT_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL, iTer-
  mId, iSyncId);
2129        else
2130           WriteZString(pECB, MakePaymentForm(iTermId,
  iSyncId, FALSE) );
2131
2132        return;
2133    }
2134
2135    /* FUNCTION: void ProcessOrderStatus-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2136     *
2137     * PURPOSE: This function gets and validates the
  input data from the Order Status
2138     *          form filling in the required input vari-
  ables. It then calls the
2139     *          SQLOrderStatus transaction, constructs
  the output form and writes it
2140     *          back to client browser.
2141     *
2142     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
  passed in structure pointer from inetsrv.
2143     *              int                 iTermId client
  browser terminal id
2144     *              int                 iSyncId client
  browser sync id
2145     *
2146     * RETURNS:     None
2147     *
2148     * COMMENTS:    None
2149     *
2150     */
2151
2152    static void ProcessOrderStatus-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2153    {
2154        int         iRc;
2155        int         iError;
2156        PECBINFO    pEcbInfo;
2157
2158
2159        memset(&Term.pClientData[iTermId].orderStatus-
  Data, 0, sizeof(ORDER_STATUS_DATA));
2160
2161        Term.pClientData[iTermId].orderStatusData.w_id
  = Term.pClientData[iTermId].w_id;
2162
2163        if ( (iError=GetOrderStatusData(pECB->lpszQue-
  ryString, &Term.pClientData[iTermId].orderStatusData))
  != ERR_SUCCESS )
2164        {
2165           ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL,
  NULL, iTermId, iSyncId);
2166            return;
2167        }
2168
2169        iRc = SQLOrderStatus(pECB, iTermId, iSyncId,
  Term.pClientData[iTermId].dbproc, &Term.pClient-
  Data[iTermId].orderStatusData, iDeadlockRetry);
2170
2171    #ifdef USE_ODBC
2172        #if (ODBCVER >= 0x0300)
2173            if ( bConnectionPooling && iRc != -3 )
2174                SQLDisconnect(Term.pClientData[iTer-
  mId].dbproc->hdbc);
2175        #endif
2176    #endif
2177
2178        if ((pEcbInfo = SQLGetECB(Term.pClient-
  Data[iTermId].dbproc)) &&
2179            pEcbInfo->bFailed)
2180            return;
2181
2182        if ( iRc == 0 )
2183            ErrorMessage(pECB, ERR_NOSUCH_CUSTOMER,
  ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
2184        else if ( iRc < 0 )
2185            ErrorMessage(pECB,
  ERR_ORDER_STATUS_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2186        else
2187            WriteZString(pECB, MakeOrderStatus-
  Form(iTermId, iSyncId, FALSE) );
2188
2189        return;
2190    }
2191
2192    /* FUNCTION: void ProcessDelivery-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2193     *
2194     * PURPOSE: This function gets and validates the
  input data from the delivery form
2195     *          filling in the required input variables.
  It then calls the PostDeliveryInfo
2196     *           Api, The client is then informed that
  the transaction has been posted.
2197     *
2198     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
  passed in structure pointer from inetsrv.
2199     *              int                 iTermId client
  browser terminal id
2200     *              int                 iSyncId clinet
  browser sync id
```

```
2201     *
2202     * RETURNS:     None
2203     *
2204     * COMMENTS:    None
2205     *
2206     */
2207
2208    static void ProcessDelivery-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2209    {
2210        char    szTmp[26];
2211        BOOL    bSuccess;
2212
2213       memset(&Term.pClientData[iTermId].deliveryData,
  0, sizeof(DELIVERY_DATA));
2214
2215        Term.pClientData[iTermId].deliveryData.w_id =
  Term.pClientData[iTermId].w_id;
2216
2217        if ( !GetKeyValue(pECB->lpszQueryString,
  "OCD*", szTmp, sizeof(szTmp)) )
2218        {
2219            ErrorMessage(pECB,
  ERR_DELIVERY_MISSING_OCD_KEY, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2220            return;
2221        }
2222
2223        if ( !IsNumeric(szTmp) )
2224        {
2225            ErrorMessage(pECB,
  ERR_DELIVERY_CARRIER_INVALID, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2226            return;
2227        }
2228
2229        Term.pClientData[iTermId].delivery-
  Data.o_carrier_id = atoi(szTmp);
2230
2231
2232        if ( Term.pClientData[iTermId].delivery-
  Data.o_carrier_id > 10 || Term.pClientData[iTer-
  mId].deliveryData.o_carrier_id < 1 )
2233        {
2234            ErrorMessage(pECB,
  ERR_DELIVERY_CARRIER_ID_RANGE, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2235            return;
2236        }
2237
2238        //post delivery info
2239        if ( PostDeliveryInfo(Term.pClientData[iTer-
  mId].deliveryData.w_id, Term.pClientData[iTermId].deliv-
  eryData.o_carrier_id) )
2240        {
2241            strcpy(Term.pClientData[iTermId].delivery-
  Data.execution_status, "Delivery Post Failed");
2242            bSuccess = FALSE;
2243        }
2244        else
2245        {
2246            strcpy(Term.pClientData[iTermId].delivery-
  Data.execution_status, "Delivery has been queued.");
2247            bSuccess = TRUE;
2248        }
2249
2250        WriteZString(pECB, MakeDeliveryForm(iTermId,
  iSyncId, FALSE, bSuccess) );
2251
2252        return;
2253    }
2254
2255    /* FUNCTION: void ProcessStockLevel-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2256     *
2257     * PURPOSE: This function gets and validates the
  input data from the Stock Level
2258     *          form filling in the required input vari-
  ables. It then calls the
2259     *          SQLStockLevel transaction, constructs
  the output form and writes it
2260     *          back to client browser.
2261     *
2262     * ARGUMENTS:   EXTENSION_CONTROL_BLOCK *pECB
  passed in structure pointer from inetsrv.
2263     *              int                   iTermId client
  browser terminal id
2264     *              int                   iSyncId client
  browser sync id
2265     *
2266     * RETURNS:     None
2267     *
2268     * COMMENTS:    None
2269     *
2270     */
2271
2272    static void ProcessStockLevel-
  Form(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
  iSyncId)
2273    {
2274        char        szTmp[26];
2275        int         iRc;
2276        PECBINFO    pEcbInfo;
2277
2278
2279        memset(&Term.pClientData[iTermId].stockLevel-
  Data, 0, sizeof(STOCK_LEVEL_DATA));
2280
2281       Term.pClientData[iTermId].stockLevelData.w_id =
  Term.pClientData[iTermId].w_id;
2282       Term.pClientData[iTermId].stockLevelData.d_id =
  Term.pClientData[iTermId].d_id;
2283
2284       if ( !GetKeyValue(pECB->lpszQueryString, "TT*",
  szTmp, sizeof(szTmp)) )
2285        {
2286            ErrorMessage(pECB,
  ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY, ERR_TYPE_WEBDLL,
  NULL, iTermId, iSyncId);
2287            return;
2288        }
2289
2290        if ( !IsNumeric(szTmp) )
2291        {
2292            ErrorMessage(pECB,
  ERR_STOCKLEVEL_THRESHOLD_INVALID, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2293            return;
2294        }
2295
2296        Term.pClientData[iTermId].stockLevel-
  Data.thresh_hold = atoi(szTmp);
2297
2298        if ( Term.pClientData[iTermId].stockLevel-
  Data.thresh_hold >= 100 || Term.pClientData[iTer-
  mId].stockLevelData.thresh_hold < 0 )
2299        {
2300            ErrorMessage(pECB,
  ERR_STOCKLEVEL_THRESHOLD_RANGE, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2301            return;
2302        }
2303
2304        iRc = SQLStockLevel(pECB, iTermId, iSyncId,
  Term.pClientData[iTermId].dbproc, &Term.pClient-
  Data[iTermId].stockLevelData, iDeadlockRetry);
2305
2306    #ifdef USE_ODBC
2307        #if (ODBCVER >= 0x0300)
2308            if ( bConnectionPooling && iRc != -3 )
2309                SQLDisconnect(Term.pClientData[iTer-
  mId].dbproc->hdbc);
```

```
2310        #endif
2311    #endif
2312
2313        if ((pEcbInfo = SQLGetECB(Term.pClient-
  Data[iTermId].dbproc)) &&
2314            pEcbInfo->bFailed)
2315            return;
2316
2317        if ( iRc )
2318            ErrorMessage(pECB,
  ERR_STOCKLEVEL_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL,
  iTermId, iSyncId);
2319        else
2320            WriteZString(pECB, MakeStockLevelForm(iTer-
  mId, iSyncId, FALSE) );
2321
2322        return;
2323    }
2324
2325    /* FUNCTION: int GetNewOrderData(LPSTR lpszQue-
  ryString, NEW_ORDER_DATA *pNewOrderData)
2326     *
2327     * PURPOSE: This function extracts and validates
  the new order form data from an http command string.
2328     *
2329     * ARGUMENTS:   LPSTR          lpszQueryString
  client browser http command string
2330     *             NEW_ORDER_DATA  *pNewOrderData
  pointer to new order data structure
2331     *
2332     * RETURNS:     int
  error code indicating reason for failure
2333     *            ERR_SUCCESS                      new
  order input data successfully parsed
2334     *
2335     *
2336     * COMMENTS:    None
2337     *
2338     */
2339
2340    static int GetNewOrderData(LPSTR lpszQueryString,
  NEW_ORDER_DATA *pNewOrderData)
2341    {
2342        char    szTmp[26];
2343        char    szKey[26];
2344        int     i;
2345        short   items;
2346        BOOL    bCheck;
2347
2348        if ( !GetKeyValue(lpszQueryString, "DID*",
  szTmp, sizeof(szTmp)) )
2349            return ERR_NEWORDER_FORM_MISSING_DID;
2350
2351        if ( !IsNumeric(szTmp) )
2352            return ERR_NEWORDER_DISTRICT_INVALID;
2353
2354        pNewOrderData->d_id = atoi(szTmp);
2355
2356        if ( !GetKeyValue(lpszQueryString, "CID*",
  szTmp, sizeof(szTmp)) )
2357            return ERR_NEWORDER_CUSTOMER_KEY;
2358
2359        if ( !IsNumeric(szTmp) )
2360            return ERR_NEWORDER_CUSTOMER_INVALID;
2361        pNewOrderData->c_id = atoi(szTmp);
2362
2363        bCheck = FALSE;
2364        for(i=0, items=0; i<15; i++)
2365        {
2366            wsprintf(szKey, "IID%2.2d*", i);
2367            if ( !GetKeyValue(lpszQueryString, szKey,
  szTmp, sizeof(szTmp)) )
2368                return ERR_NEWORDER_MISSING_IID_KEY;
2369            if ( szTmp[0] )
2370            {
2371                //if blank lines between item ids
2372                if ( bCheck )
2373                    return
  ERR_NEWORDER_ITEM_BLANK_LINES;
2374                if ( !IsNumeric(szTmp) )
2375                    return ERR_NEWORDER_ITEMID_INVALID;
2376                pNewOrderData->Ol[i].ol_i_id =
  atoi(szTmp);
2377
2378                wsprintf(szKey, "SP%2.2d*", i);
2379                if ( !GetKeyValue(lpszQueryString,
  szKey, szTmp, sizeof(szTmp)) )
2380                    return
  ERR_NEWORDER_MISSING_SUPPW_KEY;
2381                if ( !IsNumeric(szTmp) )
2382                    return ERR_NEWORDER_SUPPW_INVALID;
2383                pNewOrderData->Ol[i].ol_supply_w_id =
  (short)atoi(szTmp);
2384
2385                wsprintf(szKey, "Qty%2.2d*", i);
2386                if ( !GetKeyValue(lpszQueryString,
  szKey, szTmp, sizeof(szTmp)) )
2387                    return ERR_NEWORDER_MISSING_QTY_KEY;
2388
2389                if ( !IsNumeric(szTmp) )
2390                    return ERR_NEWORDER_QTY_INVALID;
2391
2392                pNewOrderData->Ol[i].ol_quantity =
  atoi(szTmp);
2393                items++;
2394
2395                if ( pNewOrderData->Ol[i].ol_i_id >=
  1000000 || pNewOrderData->Ol[i].ol_i_id < 1 )
2396                    return ERR_NEWORDER_ITEMID_RANGE;
2397              if ( pNewOrderData->Ol[i].ol_quantity >=
  100 || pNewOrderData->Ol[i].ol_quantity < 1 )
2398                    return ERR_NEWORDER_QTY_RANGE;
2399            }
2400            else
2401            {
2402                wsprintf(szKey, "SP%2.2d*", i);
2403                if ( !GetKeyValue(lpszQueryString,
  szKey, szTmp, sizeof(szTmp)) )
2404                    return ERR_NEWORDER_MISSING_QTY_KEY;
2405
2406                if ( szTmp[0] )
2407                    return
  ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;
2408
2409                wsprintf(szKey, "Qty%2.2d*", i);
2410                if ( !GetKeyValue(lpszQueryString,
  szKey, szTmp, sizeof(szTmp)) )
2411                    return ERR_NEWORDER_MISSING_QTY_KEY;
2412
2413                if ( szTmp[0] )
2414                    return
  ERR_NEWORDER_QTY_WITHOUT_ITEMID;
2415
2416                bCheck = TRUE;
2417            }
2418        }
2419        if ( items == 0 )
2420            return ERR_NEWORDER_NOITEMS_ENTERED;
2421
2422        pNewOrderData->o_ol_cnt = items;
2423
2424        return ERR_SUCCESS;
2425    }
2426
2427    /* FUNCTION: int GetPaymentData(LPSTR lpszQue-
  ryString, PAYMENT_DATA *pPaymentData)
2428     *
2429     * PURPOSE: This function extracts and validates
  the payment form data from an http command string.
2430     *
2431     * ARGUMENTS:   LPSTR          lpszQueryString
  client browser http command string
2432     *             PAYMENT_DATA    *pPaymentData
  pointer to payment data structure
```

```
2433    *
2434    * RETURNS:         int
  error code indicating reason for failure
2435    *                ERR_SUCCESS                    all
  input data successfully parsed
2436    *
2437    * COMMENTS:    None
2438    *
2439    */
2440
2441   static int GetPaymentData(LPSTR lpszQueryString,
  PAYMENT_DATA *pPaymentData)
2442   {
2443       char    szTmp[26];
2444       char    *ptr;
2445
2446       if ( !GetKeyValue(lpszQueryString, "DID*",
  szTmp, sizeof(szTmp)) )
2447           return ERR_PAYMENT_MISSING_DID_KEY;
2448       if ( !IsNumeric(szTmp) )
2449           return ERR_PAYMENT_DISTRICT_INVALID;
2450       pPaymentData->d_id = atoi(szTmp);
2451
2452       if ( !GetKeyValue(lpszQueryString, "CID*",
  szTmp, sizeof(szTmp)) )
2453           return ERR_PAYMENT_MISSING_CID_KEY;
2454
2455       if ( szTmp[0] && !IsNumeric(szTmp) )
2456           return ERR_PAYMENT_CUSTOMER_INVALID;
2457
2458
2459       pPaymentData->c_id = atoi(szTmp);
2460
2461       if ( szTmp[0] == 0 )
2462       {
2463           if ( !GetKeyValue(lpszQueryString, "CLT*",
  szTmp, sizeof(szTmp)) )
2464               return ERR_PAYMENT_MISSING_CLT;
2465           _strupr( szTmp );
2466
2467           strcpy(pPaymentData->c_last, szTmp);
2468           if ( strlen(pPaymentData->c_last) > 16 )
2469               return ERR_PAYMENT_LAST_NAME_TO_LONG;
2470       }
2471       else
2472       {
2473           if ( !GetKeyValue(lpszQueryString, "CLT*",
  szTmp, sizeof(szTmp)) )
2474               return ERR_PAYMENT_MISSING_CLT_KEY;
2475           if ( szTmp[0] )
2476               return ERR_PAYMENT_CID_AND_CLT;
2477       }
2478
2479       if ( !GetKeyValue(lpszQueryString, "CDI*",
  szTmp, sizeof(szTmp)) )
2480           return ERR_PAYMENT_MISSING_CDI_KEY;
2481       if ( !IsNumeric(szTmp) )
2482           return ERR_PAYMENT_CDI_INVALID;
2483       pPaymentData->c_d_id = atoi(szTmp);
2484
2485       if ( !GetKeyValue(lpszQueryString, "CWI*",
  szTmp, sizeof(szTmp)) )
2486           return ERR_PAYMENT_MISSING_CWI_KEY;
2487
2488       if ( !IsNumeric(szTmp) )
2489           return ERR_PAYMENT_CWI_INVALID;
2490
2491       pPaymentData->c_w_id = atoi(szTmp);
2492
2493
2494       if ( !GetKeyValue(lpszQueryString, "HAM*",
  szTmp, sizeof(szTmp)) )
2495           return ERR_PAYMENT_MISSING_HAM_KEY;
2496
2497       ptr = szTmp;
2498
2499       while( *ptr )
2500       {
2501           if ( *ptr == '.' )
2502           {
2503               ptr++;
2504               if ( !*ptr )
2505                   break;
2506               if ( *ptr < '0' || *ptr > '9' )
2507                   return ERR_PAYMENT_HAM_INVALID;
2508               ptr++;
2509               if ( !*ptr )
2510                   break;
2511               if ( *ptr < '0' || *ptr > '9' )
2512                   return ERR_PAYMENT_HAM_INVALID;
2513               if ( !*ptr )
2514                   return ERR_PAYMENT_HAM_INVALID;
2515           }
2516           else if ( *ptr < '0' || *ptr > '9' )
2517                   return ERR_PAYMENT_HAM_INVALID;
2518           ptr++;
2519       }
2520
2521       pPaymentData->h_amount = atof(szTmp);
2522      if ( pPaymentData->h_amount >= 10000.00 || pPay-
  mentData->h_amount < 0 )
2523           return ERR_PAYMENT_HAM_RANGE;
2524
2525       return ERR_SUCCESS;
2526   }
2527
2528   /* FUNCTION: int GetOrderStatusData(LPSTR lpszQue-
  ryString, ORDER_STATUS_DATA *pOrderStatusData)
2529    *
2530    * PURPOSE: This function extracts and validates
  the payment form data from an http command string.
2531    *
2532    * ARGUMENTS:  LPSTR            lpszQueryString
  client browser http command string
2533    *             ORDER_STATUS_DATA   *pOrderStatus-
  Data   pointer to order status data structure
2534    *
2535    * RETURNS:     int
  error code indicating reason for failure
2536    *             ERR_SUCCESS
  successfully parsed all required input data
2537    *
2538    * COMMENTS:    None
2539    *
2540    */
2541   static int GetOrderStatusData(LPSTR lpszQue-
  ryString, ORDER_STATUS_DATA *pOrderStatusData)
2542   {
2543       char    szTmp[26];
2544
2545       if ( !GetKeyValue(lpszQueryString, "DID*",
  szTmp, sizeof(szTmp)) )
2546           return ERR_ORDERSTATUS_MISSING_DID_KEY;
2547       if ( !IsNumeric(szTmp) )
2548           return ERR_ORDERSTATUS_DID_INVALID;
2549       pOrderStatusData->d_id = atoi(szTmp);
2550
2551       if ( !GetKeyValue(lpszQueryString, "CID*",
  szTmp, sizeof(szTmp)) )
2552           return ERR_ORDERSTATUS_MISSING_CID_KEY;
2553
2554       if ( szTmp[0] == 0 )
2555       {
2556           pOrderStatusData->c_id = 0;
2557           if ( !GetKeyValue(lpszQueryString, "CLT*",
  szTmp, sizeof(szTmp)) )
2558               return ERR_ORDERSTATUS_MISSING_CLT_KEY;
2559           _strupr( szTmp );
2560           strcpy(pOrderStatusData->c_last, szTmp);
2561          if ( strlen(pOrderStatusData->c_last) > 16 )
2562               return ERR_ORDERSTATUS_CLT_RANGE;
2563       }
2564       else
2565       {
```

```
2566            if ( !IsNumeric(szTmp) )
2567                return ERR_ORDERSTATUS_CID_INVALID;
2568            pOrderStatusData->c_id = atoi(szTmp);
2569            if ( !GetKeyValue(lpszQueryString, "CLT*",
  szTmp, sizeof(szTmp)) )
2570                return ERR_ORDERSTATUS_MISSING_CLT_KEY;
2571            if ( szTmp[0] )
2572                return ERR_ORDERSTATUS_CID_AND_CLT;
2573        }
2574
2575        return ERR_SUCCESS;
2576    }
2577
2578    /* FUNCTION: BOOL ReadRegistrySettings(void)
2579     *
2580     * PURPOSE: This function reads the NT registry for
  startup parameters. There parameters are
2581     *          under the TPCC key.
2582     *
2583     * ARGUMENTS:   None
2584     *
2585     * RETURNS:     None
2586     *
2587     * COMMENTS:    This function also sets up required
  operation variables to their default value
2588     *              so if registry is not setup the
  default values will be used.
2589     *
2590     */
2591
2592    static BOOL ReadRegistrySettings(void)
2593    {
2594        HKEY    hKey;
2595        DWORD   size;
2596        DWORD   type;
2597        char    szTmp[256];
2598
2599        bLog            = FALSE;
2600        iMaxWareHouses  = 500;
2601        iThreads        = 5;
2602        iQSlotts        = 3000;
2603        iDelayMs        = 100;
2604        iDeadlockRetry  = (short)3;
2605        strcpy(szTpccLogPath, "tpcclog.");
2606
2607    #ifdef USE_ODBC
2608        bConnectionPooling = FALSE;
2609    #endif
2610
2611        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFT-
  WARE\\Microsoft\\TPCC", 0, KEY_READ, &hKey) !=
  ERROR_SUCCESS )
2612            return TRUE;
2613        size = sizeof(szTmp);
2614
2615        if ( RegQueryValueEx(hKey, "PATH", 0, &type,
  szTmp, &size) == ERROR_SUCCESS )
2616        {
2617            strcpy(szTpccLogPath, szTmp);
2618            strcat(szTpccLogPath, "tpcclog.");
2619            strcpy(szErrorLogPath, szTmp);
2620            strcat(szErrorLogPath, "tpccerr.");
2621        }
2622
2623        size = sizeof(szTmp);
2624        if ( RegQueryValueEx(hKey, "LOG", 0, &type,
  szTmp, &size) == ERROR_SUCCESS )
2625        {
2626            if ( !stricmp(szTmp, "ON") )
2627                bLog = TRUE;
2628        }
2629
2630        size = sizeof(szTmp);
2631       if ( RegQueryValueEx(hKey, "MaximumWarehouses",
  0, &type, szTmp, &size) == ERROR_SUCCESS )
2632        {
2633            iMaxWareHouses = atoi(szTmp);
2634            if ( iMaxWareHouses == 0 )
2635                iMaxWareHouses = 500;
2636        }
2637
2638        size = sizeof(szTmp);
2639        if ( RegQueryValueEx(hKey, "NumberOfDeliveryTh-
  reads", 0, &type, szTmp, &size) == ERROR_SUCCESS )
2640            iThreads = atoi(szTmp);
2641        if ( !iThreads )
2642            iThreads = 5;
2643
2644        size = sizeof(szTmp);
2645        if ( RegQueryValueEx(hKey, "QueueSlotts", 0,
  &type, szTmp, &size) == ERROR_SUCCESS )
2646            iQSlotts = atoi(szTmp);
2647            if ( !iQSlotts )
2648                iQSlotts = 3000;
2649
2650        size = sizeof(szTmp);
2651        if ( RegQueryValueEx(hKey, "BackoffDelay", 0,
  &type, szTmp, &size) == ERROR_SUCCESS )
2652            iDelayMs = atoi(szTmp);
2653            if ( !iDelayMs )
2654                iDelayMs = 100;
2655
2656        size = sizeof(szTmp);
2657        if ( RegQueryValueEx(hKey, "DeadlockRetry", 0,
  &type, szTmp, &size) == ERROR_SUCCESS )
2658            iDeadlockRetry = (short)atoi(szTmp);
2659            if ( !iDeadlockRetry )
2660                iDeadlockRetry = (short)3;
2661
2662        size = sizeof(szTmp);
2663       if ( RegQueryValueEx(hKey, "MaxConnections", 0,
  &type, szTmp, &size) == ERROR_SUCCESS )
2664            iMaxConnections = (short)atoi(szTmp);
2665            if ( !iMaxConnections )
2666                iMaxConnections = (short)25;
2667
2668    #ifdef USE_ODBC
2669        #if (ODBCVER >= 0x0300)
2670            size = sizeof(szTmp);
2671            if ( RegQueryValueEx(hKey, "ConnectionPool-
  ing", 0, &type, szTmp, &size) == ERROR_SUCCESS )
2672                if ( !stricmp(szTmp, "ON") )
2673                    bConnectionPooling = TRUE;
2674
2675            iConnectDelay = 500;
2676            size = sizeof(szTmp);
2677            if ( RegQueryValueEx(hKey, "ConnectionPool-
  RetryTime", 0, &type, szTmp, &size) == ERROR_SUCCESS )
2678                iConnectDelay = atoi(szTmp);
2679            if ( !iConnectDelay )
2680                iConnectDelay = 500;
2681
2682        #endif
2683    #endif
2684
2685        RegCloseKey(hKey);
2686
2687        return FALSE;
2688
2689    }
2690
2691    /* FUNCTION: BOOL PostDeliveryInfo(short w_id,
  short o_carrier_id)
2692     *
2693     * PURPOSE: This function writes the delivery
  information to the delivery pipe. The information is
2694     *          sent as a long.
2695     *
2696     * ARGUMENTS:   short           w_id            ware-
  house id
2697     *              short           o_carrier_id    car-
  rier id
2698     *
2699     * RETURNS:     BOOL    FALSE   delivery information
```

```
   posted successfully
2700     *                       TRUE    error cannot post
   delivery info
2701     *
2702     * COMMENTS:    The pipe is initially created with
   16K buffer size this should allow for
2703     *              up to 4096 deliveries to be queued
   before an overflow condition would
2704     *              occur. The only reason that an over-
   flow would occur is if the delivery
2705     *              application stopped listening while
   deliveries were being posted.
2706     *
2707     */
2708
2709    static BOOL PostDeliveryInfo(short w_id, short
   o_carrier_id)
2710    {
2711        DELIVERY_TRANSACTION    deliveryTransaction;
2712        int                     d;
2713        int                     i;
2714
2715        GetLocalTime(&deliveryTransaction.queue);
2716
2717        deliveryTransaction.w_id            =   w_id;
2718        deliveryTransaction.o_carrier_id    =
   o_carrier_id;
2719
2720        for(i=0; i<4; i++)
2721        {
2722           if ( WriteFile(hPipe, &deliveryTransaction,
   sizeof(deliveryTransaction), &d, NULL) )
2723               return FALSE;
2724
2725           if ( GetLastError() != ERROR_PIPE_BUSY )
   //ERROR_PIPE_LISTENING
2726               return TRUE;
2727        }
2728
2729        return TRUE;
2730    }
2731
2732    /* FUNCTION: BOOL IsNumeric(char *ptr)
2733     *
2734     * PURPOSE: This function determines if a string is
   numeric. It fails if any characters other
2735     *          than numeric and null terminator are
   present.
2736     *
2737     * ARGUMENTS:   char            *ptr    pointer to
   string to check.
2738     *
2739     * RETURNS:     BOOL    FALSE   if string is not all
   numeric
2740     *                      TRUE    if string contains
   only numeric characters i.e. '0' - '9'
2741     *
2742     * COMMENTS:    None
2743     *
2744     */
2745
2746    static BOOL IsNumeric(char *ptr)
2747    {
2748        if ( *ptr == 0 )
2749            return FALSE;
2750
2751        while( *ptr && isdigit(*ptr) )
2752            ptr++;
2753        return ( !*ptr );
2754    }
2755
2756    /* FUNCTION: void FormatHTMLString(char *szBuff,
   int iLen, char *szStr)
2757     *
2758     * PURPOSE: This function Handles translation of
   HTML specific character field data
2759     *          when an HTML output form is generated.
2760     *
2761     * ARGUMENTS:   char    *szBuff Returned string
   information
2762     *              char    *szStr  input string to be
   formatted.
2763     *              int     iLen    Length of returned
   string
2764     *
2765     * RETURNS:     none
2766     *
2767     * COMMENTS:    The length paramter is the absolute
   length of the returned string in
2768     *              HTML characters. For example the
   input string > would be returned as
2769     *              &gt; which would be counted as 1
   character.If the number of input
2770     *              characters is less than the iLen
   parameter spaces are appended to
2771     *             the end of the string to ensure that
   at least iLen characters are
2772     *              returned in the szBuff parameter.
2773     *
2774     */
2775
2776    static void FormatHTMLString(char *szBuff, char
   *szStr, int iLen)
2777    {
2778        while( iLen && *szStr )
2779        {
2780            switch( *szStr )
2781            {
2782                case '>':
2783                    *szBuff++ = '&';
2784                    *szBuff++ = 'g';
2785                    *szBuff++ = 't';
2786                    *szBuff++ = ';';
2787                    szStr++;
2788                    break;
2789                case '<':
2790                    *szBuff++ = '&';
2791                    *szBuff++ = 'l';
2792                    *szBuff++ = 't';
2793                    *szBuff++ = ';';
2794                    szStr++;
2795                    break;
2796                case '&':
2797                    *szBuff++ = '&';
2798                    *szBuff++ = 'a';
2799                    *szBuff++ = 'm';
2800                    *szBuff++ = 'p';
2801                    *szBuff++ = ';';
2802                    szStr++;
2803                    break;
2804                case '\"':
2805                    *szBuff++ = '&';
2806                    *szBuff++ = 'q';
2807                    *szBuff++ = 'u';
2808                    *szBuff++ = 'o';
2809                    *szBuff++ = 't';
2810                    *szBuff++ = ';';
2811                    szStr++;
2812                    break;
2813                default:
2814                    *szBuff++ = *szStr++;
2815                    break;
2816            }
2817            iLen--;
2818        }
2819        while( iLen-- )
2820            *szBuff++ = ' ';
2821
2822        *szBuff = 0;
2823
2824        return;
2825    }
```

tpcc.h

```
#ifndef TPCC_H_INCLUDED
#define TPCC_H_INCLUDED


extern char szErrorLogPath[];

#ifdef TUX
#include "tpcc_tux.h"
#else
#include <httpext.h>
#include "tpcc_real.h"
#endif

#endif
```

tpcc.mak

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

# TARGTYPE "Win32 (x86) External Target" 0x0106

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

!IF "$(CFG)" != "tpcc - Win32 Release" && "$(CFG)" != "tpcc - Win32 Debug" &&\
 "$(CFG)" != "tux_client - Win32 Release" && "$(CFG)" !=\
 "tux_client - Win32 Debug" && "$(CFG)" != "tux_server - Win32 Release" &&\
 "$(CFG)" != "tux_server - Win32 Debug" && "$(CFG)" != "dll - Win32 Release" &&\
 "$(CFG)" != "dll - 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 Release"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "tpcc - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "tpcc - Win32 Debug" (based on "Win32 (x86) External Target")
!MESSAGE "tux_client - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "tux_client - Win32 Debug" (based on "Win32 (x86) External Target")
!MESSAGE "tux_server - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "tux_server - Win32 Debug" (based on "Win32 (x86) External Target")
!MESSAGE "dll - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "dll - Win32 Debug" (based on "Win32 (x86) External Target")
!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"

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

# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP BASE Cmd_Line "NMAKE /f tpcc.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tpcc.exe"
# PROP BASE Bsc_Name "tpcc.bsc"
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
# PROP Cmd_Line "NMAKE /f tpcc.mak"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tpcc.exe"
# PROP Bsc_Name "tpcc.bsc"
OUTDIR=.\Release
INTDIR=.\Release

ALL : "dll - Win32 Release" "tux_server - Win32 Release"\
 "tux_client - Win32 Release"

CLEAN :
    -@erase

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

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

# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP BASE Cmd_Line "NMAKE /f tpcc.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tpcc.exe"
# PROP BASE Bsc_Name "tpcc.bsc"
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
# PROP Cmd_Line "NMAKE /f tpcc.mak"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tpcc.exe"
# PROP Bsc_Name "tpcc.bsc"
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : "dll - Win32 Debug" "tux_server - Win32 Debug" "tux_client - Win32 Debug"\


CLEAN :
    -@erase

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

!ELSEIF  "$(CFG)" == "tux_client - Win32 Release"

# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "tux_client\Release"
# PROP BASE Intermediate_Dir "tux_client\Release"
# PROP BASE Target_Dir "tux_client"
# PROP BASE Cmd_Line "NMAKE /f tux_client.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tux_client\tux_client.exe"
# PROP BASE Bsc_Name "tux_client\tux_client.bsc"
```

```
115    # PROP Use_Debug_Libraries 0
116    # PROP Output_Dir "tux_client\Release"
117    # PROP Intermediate_Dir "tux_client\Release"
118    # PROP Target_Dir "tux_client"
119    # PROP Cmd_Line "NMAKE CFG=Release /f
 tux_client.mak"
120    # PROP Rebuild_Opt "/a"
121    # PROP Target_File "tux_client\tux_client.exe"
122    # PROP Bsc_Name "tux_client\tux_client.bsc"
123    OUTDIR=.\tux_client\Release
124    INTDIR=.\tux_client\Release
125
126    ALL :
127
128    CLEAN :
129        -@erase
130
131    "$(OUTDIR)" :
132        if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUT-
 DIR)"
133
134    !ELSEIF  "$(CFG)" == "tux_client - Win32 Debug"
135
136    # PROP BASE Use_Debug_Libraries 1
137    # PROP BASE Output_Dir "tux_client\Debug"
138    # PROP BASE Intermediate_Dir "tux_client\Debug"
139    # PROP BASE Target_Dir "tux_client"
140    # PROP BASE Cmd_Line "NMAKE /f tux_client.mak"
141    # PROP BASE Rebuild_Opt "/a"
142    # PROP BASE Target_File "tux_client\tux_client.exe"
143    # PROP BASE Bsc_Name "tux_client\tux_client.bsc"
144    # PROP Use_Debug_Libraries 1
145    # PROP Output_Dir "tux_client\Debug"
146    # PROP Intermediate_Dir "tux_client\Debug"
147    # PROP Target_Dir "tux_client"
148    # PROP Cmd_Line "NMAKE CFG=Debug /f tux_client.mak"
149    # PROP Rebuild_Opt "/a"
150    # PROP Target_File "tux_client\tux_client.exe"
151    # PROP Bsc_Name "tux_client\tux_client.bsc"
152    OUTDIR=.\tux_client\Debug
153    INTDIR=.\tux_client\Debug
154
155    ALL :
156
157    CLEAN :
158        -@erase
159
160    "$(OUTDIR)" :
161        if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUT-
 DIR)"
162
163    !ELSEIF  "$(CFG)" == "tux_server - Win32 Release"
164
165    # PROP BASE Use_Debug_Libraries 0
166    # PROP BASE Output_Dir "tux_server\Release"
167    # PROP BASE Intermediate_Dir "tux_server\Release"
168    # PROP BASE Target_Dir "tux_server"
169    # PROP BASE Cmd_Line "NMAKE /f tux_server.mak"
170    # PROP BASE Rebuild_Opt "/a"
171    # PROP BASE Target_File "tux_server\tux_server.exe"
172    # PROP BASE Bsc_Name "tux_server\tux_server.bsc"
173    # PROP Use_Debug_Libraries 0
174    # PROP Output_Dir "tux_server\Release"
175    # PROP Intermediate_Dir "tux_server\Release"
176    # PROP Target_Dir "tux_server"
177    # PROP Cmd_Line "NMAKE CFG=Release /f
 tux_server.mak"
178    # PROP Rebuild_Opt "/a"
179    # PROP Target_File
 "c:\temp\mckee\tpcc\tux_server\tpcc.exe"
180    # PROP Bsc_Name ""
181    OUTDIR=.\tux_server\Release
182    INTDIR=.\tux_server\Release
183
184    ALL :
185
186    CLEAN :
187        -@erase
188
189    "$(OUTDIR)" :
190        if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUT-
 DIR)"
191
192    !ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
193
194    # PROP BASE Use_Debug_Libraries 1
195    # PROP BASE Output_Dir "tux_server\Debug"
196    # PROP BASE Intermediate_Dir "tux_server\Debug"
197    # PROP BASE Target_Dir "tux_server"
198    # PROP BASE Cmd_Line "NMAKE /f tux_server.mak"
199    # PROP BASE Rebuild_Opt "/a"
200    # PROP BASE Target_File "tux_server\tux_server.exe"
201    # PROP BASE Bsc_Name "tux_server\tux_server.bsc"
202    # PROP Use_Debug_Libraries 1
203    # PROP Output_Dir "tux_server\Debug"
204    # PROP Intermediate_Dir "tux_server\Debug"
205    # PROP Target_Dir "tux_server"
206    # PROP Cmd_Line "NMAKE CFG=Debug /f tux_server.mak"
207    # PROP Rebuild_Opt "/a"
208    # PROP Target_File
 "c:\temp\mckee\tpcc\tux_server\tpcc.exe"
209    # PROP Bsc_Name ""
210    OUTDIR=.\tux_server\Debug
211    INTDIR=.\tux_server\Debug
212
213    ALL :
214
215    CLEAN :
216        -@erase
217
218    "$(OUTDIR)" :
219        if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUT-
 DIR)"
220
221    !ELSEIF  "$(CFG)" == "dll - Win32 Release"
222
223    # PROP BASE Use_Debug_Libraries 0
224    # PROP BASE Output_Dir "dll\Release"
225    # PROP BASE Intermediate_Dir "dll\Release"
226    # PROP BASE Target_Dir "dll"
227    # PROP BASE Cmd_Line "NMAKE /f dll.mak"
228    # PROP BASE Rebuild_Opt "/a"
229    # PROP BASE Target_File "dll\dll.exe"
230    # PROP BASE Bsc_Name "dll\dll.bsc"
231    # PROP Use_Debug_Libraries 0
232    # PROP Output_Dir "dll\Release"
233    # PROP Intermediate_Dir "dll\Release"
234    # PROP Target_Dir "dll"
235    # PROP Cmd_Line "NMAKE CFG=Release /f dll.mak"
236    # PROP Rebuild_Opt "/a"
237    # PROP Target_File "dll\dll.exe"
238    # PROP Bsc_Name "dll\dll.bsc"
239    OUTDIR=.\dll\Release
240    INTDIR=.\dll\Release
241
242    ALL :
243
244    CLEAN :
245        -@erase
246
247    "$(OUTDIR)" :
248        if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUT-
 DIR)"
249
250    !ELSEIF  "$(CFG)" == "dll - Win32 Debug"
251
252    # PROP BASE Use_Debug_Libraries 1
253    # PROP BASE Output_Dir "dll\Debug"
254    # PROP BASE Intermediate_Dir "dll\Debug"
255    # PROP BASE Target_Dir "dll"
256    # PROP BASE Cmd_Line "NMAKE /f dll.mak"
257    # PROP BASE Rebuild_Opt "/a"
258    # PROP BASE Target_File "dll\dll.exe"
259    # PROP BASE Bsc_Name "dll\dll.bsc"
```

```
260    # PROP Use_Debug_Libraries 1
261    # PROP Output_Dir "dll\Debug"
262    # PROP Intermediate_Dir "dll\Debug"
263    # PROP Target_Dir "dll"
264    # PROP Cmd_Line "NMAKE CFG=Debug /f dll.mak"
265    # PROP Rebuild_Opt "/a"
266    # PROP Target_File "dll\dll.exe"
267    # PROP Bsc_Name "dll\dll.bsc"
268    OUTDIR=.\dll\Debug
269    INTDIR=.\dll\Debug
270
271    ALL :
272
273    CLEAN :
274        -@erase
275
276    "$(OUTDIR)" :
277        if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUT-
 DIR)"
278
279    !ENDIF
280
281
 ############################################################
 ########################
282    # Begin Target
283
284    # Name "tpcc - Win32 Release"
285    # Name "tpcc - Win32 Debug"
286
287    !IF  "$(CFG)" == "tpcc - Win32 Release"
288
289    ".\tpcc.exe" :
290       CD E:\home\bretm\src\win32\tpcc
291       NMAKE /f tpcc.mak
292
293    !ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
294
295    ".\tpcc.exe" :
296       CD E:\home\bretm\src\win32\tpcc
297       NMAKE /f tpcc.mak
298
299    !ENDIF
300
301
 ############################################################
 ########################
302    # Begin Project Dependency
303
304    # Project_Dep_Name "tux_client"
305
306    !IF  "$(CFG)" == "tpcc - Win32 Release"
307
308    "tux_client - Win32 Release" :
309       $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak"
 CFG="tux_client - Win32 Release"
310
311    !ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
312
313    "tux_client - Win32 Debug" :
314       $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak"
 CFG="tux_client - Win32 Debug"
315
316    !ENDIF
317
318    # End Project Dependency
319
 ############################################################
 ########################
320    # Begin Project Dependency
321
322    # Project_Dep_Name "tux_server"
323
324    !IF  "$(CFG)" == "tpcc - Win32 Release"
325
326    "tux_server - Win32 Release" :
327       $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak"
 CFG="tux_server - Win32 Release"
328
329    !ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
330
331    "tux_server - Win32 Debug" :
332       $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak"
 CFG="tux_server - Win32 Debug"
333
334    !ENDIF
335
336    # End Project Dependency
337
 ############################################################
 ########################
338    # Begin Project Dependency
339
340    # Project_Dep_Name "dll"
341
342    !IF  "$(CFG)" == "tpcc - Win32 Release"
343
344    "dll - Win32 Release" :
345       $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="dll -
 Win32 Release"
346
347    !ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
348
349    "dll - Win32 Debug" :
350       $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="dll -
 Win32 Debug"
351
352    !ENDIF
353
354    # End Project Dependency
355    # End Target
356
 ############################################################
 ########################
357    # Begin Target
358
359    # Name "tux_client - Win32 Release"
360    # Name "tux_client - Win32 Debug"
361
362    !IF  "$(CFG)" == "tux_client - Win32 Release"
363
364    ".\tux_client\tux_client.exe" :
365       CD E:\home\bretm\src\win32\tpcc\tux_client
366       NMAKE CFG=Release /f tux_client.mak
367
368    !ELSEIF  "$(CFG)" == "tux_client - Win32 Debug"
369
370    ".\tux_client\tux_client.exe" :
371       CD E:\home\bretm\src\win32\tpcc\tux_client
372       NMAKE CFG=Debug /f tux_client.mak
373
374    !ENDIF
375
376    # End Target
377
 ############################################################
 ########################
378    # Begin Target
379
380    # Name "tux_server - Win32 Release"
381    # Name "tux_server - Win32 Debug"
382
383    !IF  "$(CFG)" == "tux_server - Win32 Release"
384
385    "c:\temp\mckee\tpcc\tux_server\tpcc.exe" :
386       CD E:\home\bretm\src\win32\tpcc\tux_server
387       NMAKE CFG=Release /f tux_server.mak
388
389    !ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
390
391    "c:\temp\mckee\tpcc\tux_server\tpcc.exe" :
392       CD E:\home\bretm\src\win32\tpcc\tux_server
393       NMAKE CFG=Debug /f tux_server.mak
394
```

```
!ENDIF

# End Target
##############################################################
#######################
# Begin Target

# Name "dll - Win32 Release"
# Name "dll - Win32 Debug"

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

".\dll\dll.exe" :
   CD E:\home\bretm\src\win32\tpcc\dll
   NMAKE CFG=Release /f dll.mak

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

".\dll\dll.exe" :
   CD E:\home\bretm\src\win32\tpcc\dll
   NMAKE CFG=Debug /f dll.mak

!ENDIF


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

SOURCE=.\src\tux_trans.c

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

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

!ENDIF

# End Source File
# End Target
# End Project

##############################################################
#######################
```

tpcc_real.h

```
/*  FILE:       TPCC.H
 *              Microsoft TPC-C Kit Ver. 3.00.001
 *              Audited 08/23/96, By Francois Raab
 *
 *              Copyright Microsoft, 1996
 *
 *  PURPOSE:    Header file for ISAPI TPCC.DLL,
defines structures and functions used in the isapi
tpcc.dll.
 *  Author:     Philip Durr
 *              philipdu@Microsoft.com
 */


//VERSION RESOURCE DEFINES
#define _APS_NEXT_RESOURCE_VALUE                101
#define _APS_NEXT_COMMAND_VALUE                 40001
#define _APS_NEXT_CONTROL_VALUE                 1000
#define _APS_NEXT_SYMED_VALUE                   101


//note that the welcome form must be processed first
as terminal ids assigned here, once the
//terminal id is assigned then the forms can be pro-
cessed in any order.
#define WELCOME_FORM                        1
//beginning form no term id assigned, form id
#define MAIN_MENU_FORM                      2
//term id assigned main menu form id
#define NEW_ORDER_FORM                      3
//new order form id
#define PAYMENT_FORM                        4
//payment form id
#define DELIVERY_FORM                       5
//delivery form id
#define ORDER_STATUS_FORM                   6
//order status id
#define STOCK_LEVEL_FORM                    7
//stock level form id

//This macro is used to prevent the compiler error
unused formal parameter
#define UNUSEDPARAM(x) (x = x)


//This structure is used for posting delivery trans-
actions
typedef struct _DELIVERY_TRANSACTION
{
    SYSTEMTIME  queue;          //time delivery trans-
action queued
    short       w_id;           //delivery warehouse
    short       o_carrier_id;   //carrier id
} DELIVERY_TRANSACTION;

#ifdef USE_ODBC
    typedef struct _DBPROCESS
    {
        HDBC    hdbc;
        HSTMT   hstmt;
        int     spid;
        void    *uPtr;
    } DBPROCESS, *PDBPROCESS;

    //dblib error message return values
    #define INT_EXIT        0
    #define INT_CONTINUE    1
    #define INT_CANCEL      2
#endif

//This structure defines the data necessary to keep
distinct for each terminal or client connection.
typedef struct _CLIENTDATA
{
    int         inUse;                  //in use flag
allows client entries to be reused
    int         w_id;                   //warehouse id
assigned at welcome form
    int         d_id;                   //district id
assigned at welcome form

    PDBPROCESS  dbproc;                 //dblib con-
nection pointer
    int         spid;                   //spid assigned
from dblib
    int         iSyncId;                //syncroniza-
tion id
    int         iTickCount;             //time of last
access;
    int         iTermId;                //terminal id
of http stream connection

    char                szBuffer[4096]; //form buffer
each HTML form is built for a client in here

    NEW_ORDER_DATA      newOrderData;       //new
order form data
    PAYMENT_DATA        paymentData;        //payment
form data
    ORDER_STATUS_DATA   orderStatusData;    //order
status form data
    DELIVERY_DATA       deliveryData;       //delivery
form data
    STOCK_LEVEL_DATA    stockLevelData;     //stock
level form data
} CLIENTDATA;
```

```
typedef CLIENTDATA *PCLIENTDATA;           //pointer to client structure

//This structure is used to define the operational interface for terminal id support
typedef struct _TERM
{
    int        iAvailable;                  //total allocated terminal array entries
    int        iNext;                       //next available terminal array element
    int        iMasterSyncId;               //synchronization id
    BOOL       bInit;                       //structure has been initialized flag
    CLIENTDATA  *pClientData;
//pointer to allocated client data
    void       (*Init)(void);               //API to initialize this structure
    int        (*Allocate)(void);           //API to allocate a new terminal entry array id returned
    void       (*Restore)(void);            //API to free terminal data
    int        (*Add)(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString);  //API to add a terminal id to array, this context will
                                            //be passed from the browser to the tpcc.dll in the
                                            //TERMID= key in the HTTP string.
    void       (*Delete)(EXTENSION_CONTROL_BLOCK *pECB, int id);   //API to free resources used by a terminal array entry
} TERM;

typedef TERM *PTERM;
//pointer to terminal structure type


//function prototypes

BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved);
static void DeliveryDisconnect(void *ptr);
BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd, int *pFormId, int *pTermId, int *pSyncId);
void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...);
static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue, int iMax);
static void TermInit(void);

static void TermRestore(void);
static int TermAllocate(void);
static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString);
static void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id);
BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, char *szServer, char *szUser, char *szPassword, char *szDatabase);
BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void FormatString(char *szDest, char *szPic, char *szSrc);
static char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeMainMenuForm(int iTermId, int iSyncId);
static char *MakeWelcomeForm(void);
static char *MakeNewOrderForm(int iTermId, int iSyncId, BOOL bInput, BOOL bValid);
static char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput, BOOL bSuccess);
static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA *pNewOrderData);
static int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA *pPaymentData);
static int GetOrderStatusData(LPSTR lpszQueryString, ORDER_STATUS_DATA *pOrderStatusData);
static BOOL ReadRegistrySettings(void);
static BOOL PostDeliveryInfo(short w_id, short o_carrier_id);
static BOOL IsNumeric(char *ptr);
static void FormatHTMLString(char *szBuff, char *szStr, int iLen);

extern char szErrorLogPath[256];
extern EXTENSION_CONTROL_BLOCK  *gpECB;
```

tpcc_tux.h

```
#ifndef TPCC_TUX_H_INCLUDED
#define TPCC_TUX_H_INCLUDED
typedef char EXTENSION_CONTROL_BLOCK;

extern EXTENSION_CONTROL_BLOCK *gpECB;

typedef struct
{
    struct
    {
        char szBuffer[4096];
    } pClientData[1];

} TERM;

extern TERM Term;
```

```
   17   #endif
```

tpcc1.def

```
    1  LIBRARY TPCC.DLL
    2
    3  EXPORTS
    4
    5      GetExtensionVersion @1
    6      HttpExtensionProc   @2
```

Tpcc1.rc

```
    1  //Microsoft Developer Studio generated resource
    script.
    2  //
    3  #include "resource.h"
    4
    5  #define APSTUDIO_READONLY_SYMBOLS
    6
    //////////////////////////////////////////////////////
    ////////////////////
    7  //
    8  // Generated from the TEXTINCLUDE 2 resource.
    9  //
   10  #include "afxres.h"
   11
   12
    //////////////////////////////////////////////////////
    ////////////////////
   13  #undef APSTUDIO_READONLY_SYMBOLS
   14
   15
    //////////////////////////////////////////////////////
    ////////////////////
   16  // English (U.S.) resources
   17
   18  #if !defined(AFX_RESOURCE_DLL) ||
    defined(AFX_TARG_ENU)
   19  #ifdef _WIN32
   20  LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
   21  #pragma code_page(1252)
   22  #endif //_WIN32
   23
   24  #ifndef _MAC
   25
    //////////////////////////////////////////////////////
    ////////////////////
   26  //
   27  // Version
   28  //
   29
   30  VS_VERSION_INFO VERSIONINFO
   31   FILEVERSION 0,4,0,0
   32   PRODUCTVERSION 0,4,0,0
   33   FILEFLAGSMASK 0x3fL
   34  #ifdef _DEBUG
   35   FILEFLAGS 0x1L
   36  #else
   37   FILEFLAGS 0x0L
   38  #endif
   39   FILEOS 0x40004L
   40   FILETYPE 0x2L
   41   FILESUBTYPE 0x0L
   42  BEGIN
   43      BLOCK "StringFileInfo"
   44      BEGIN
   45          BLOCK "040904b0"
   46          BEGIN
   47              VALUE "Comments", "TPC-C HTML DLL Server
    (DBLIB)\0"
   48              VALUE "CompanyName", "Microsoft\0"
   49              VALUE "FileDescription", "TPC-C HTML DLL
    Server (DBLIB)\0"
   50              VALUE "FileVersion", "0, 4, 0, 0\0"
   51              VALUE "InternalName", "tpcc\0"
   52              VALUE "LegalCopyright", "Copyright ©
    1996\0"
   53              VALUE "OriginalFilename", "tpcc.dll\0"
   54              VALUE "ProductName", "Microsoft tpcc\0"
   55              VALUE "ProductVersion", "0, 4, 0, 0\0"
   56          END
   57      END
   58      BLOCK "VarFileInfo"
   59      BEGIN
   60          VALUE "Translation", 0x409, 1200
   61      END
   62  END
   63
   64  #endif    // !_MAC
   65
   66
   67  #ifdef APSTUDIO_INVOKED
   68
    //////////////////////////////////////////////////////
    ////////////////////
   69  //
   70  // TEXTINCLUDE
   71  //
   72
   73  1 TEXTINCLUDE DISCARDABLE
   74  BEGIN
   75      "resource.h\0"
   76  END
   77
   78  2 TEXTINCLUDE DISCARDABLE
   79  BEGIN
   80      "#include ""afxres.h""\r\n"
   81      "\0"
   82  END
   83
   84  3 TEXTINCLUDE DISCARDABLE
   85  BEGIN
   86      "\r\n"
   87      "\0"
   88  END
   89
   90  #endif    // APSTUDIO_INVOKED
   91
   92  #endif    // English (U.S.) resources
   93
    //////////////////////////////////////////////////////
    ////////////////////
   94
   95
   96
   97  #ifndef APSTUDIO_INVOKED
   98
    //////////////////////////////////////////////////////
    ////////////////////
   99  //
  100  // Generated from the TEXTINCLUDE 3 resource.
  101  //
  102
  103
  104
    //////////////////////////////////////////////////////
    ////////////////////
  105  #endif    // not APSTUDIO_INVOKED
  106
```

tpcc2.def

```
    1  LIBRARY TPCC2.DLL
    2
    3  EXPORTS
    4
    5      GetExtensionVersion @1
    6      HttpExtensionProc   @2
```

tpcc2.rc

```
1  //Microsoft Developer Studio generated resource
script.
2  //
3  #include "resource.h"
4
5  #define APSTUDIO_READONLY_SYMBOLS
6
//////////////////////////////////////////////////////
//////////////////////
7  //
8  // Generated from the TEXTINCLUDE 2 resource.
9  //
10 #include "afxres.h"
11
12
//////////////////////////////////////////////////////
//////////////////////
13 #undef APSTUDIO_READONLY_SYMBOLS
14
15
//////////////////////////////////////////////////////
//////////////////////
16 // English (U.S.) resources
17
18 #if !defined(AFX_RESOURCE_DLL) ||
defined(AFX_TARG_ENU)
19 #ifdef _WIN32
20 LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
21 #pragma code_page(1252)
22 #endif //_WIN32
23
24 #ifndef _MAC
25
//////////////////////////////////////////////////////
//////////////////////
26 //
27 // Version
28 //
29
30 VS_VERSION_INFO VERSIONINFO
31  FILEVERSION 0,4,0,0
32  PRODUCTVERSION 0,4,0,0
33  FILEFLAGSMASK 0x3fL
34 #ifdef _DEBUG
35  FILEFLAGS 0x1L
36 #else
37  FILEFLAGS 0x0L
38 #endif
39  FILEOS 0x40004L
40  FILETYPE 0x2L
41  FILESUBTYPE 0x0L
42 BEGIN
43     BLOCK "StringFileInfo"
44     BEGIN
45         BLOCK "040904b0"
46         BEGIN
47             VALUE "Comments", "TPC-C HTML DLL Server
(ODBC)\0"
48             VALUE "CompanyName", "Microsoft\0"
49             VALUE "FileDescription", "TPC-C HTML DLL
Server (ODBC)\0"
50             VALUE "FileVersion", "0, 4, 0, 0\0"
51             VALUE "InternalName", "tpcc\0"
52             VALUE "LegalCopyright", "Copyright ©
1996\0"
53             VALUE "OriginalFilename", "tpcc.dll\0"
54             VALUE "ProductName", "Microsoft tpcc\0"
55             VALUE "ProductVersion", "0, 4, 0, 0\0"
56         END
57     END
58     BLOCK "VarFileInfo"
59     BEGIN
60         VALUE "Translation", 0x409, 1200
61     END
62 END
63
64 #endif    // !_MAC
65
66
67 #ifdef APSTUDIO_INVOKED
68
//////////////////////////////////////////////////////
//////////////////////
69 //
70 // TEXTINCLUDE
71 //
72
73 1 TEXTINCLUDE DISCARDABLE
74 BEGIN
75     "resource.h\0"
76 END
77
78 2 TEXTINCLUDE DISCARDABLE
79 BEGIN
80     "#include ""afxres.h""\r\n"
81     "\0"
82 END
83
84 3 TEXTINCLUDE DISCARDABLE
85 BEGIN
86     "\r\n"
87     "\0"
88 END
89
90 #endif    // APSTUDIO_INVOKED
91
92 #endif    // English (U.S.) resources
93
//////////////////////////////////////////////////////
//////////////////////
94
95
96
97 #ifndef APSTUDIO_INVOKED
98
//////////////////////////////////////////////////////
//////////////////////
99 //
100 // Generated from the TEXTINCLUDE 3 resource.
101 //
102
103
104
//////////////////////////////////////////////////////
//////////////////////
105 #endif    // not APSTUDIO_INVOKED
106
```

trans.h

```
1  /*  FILE:       TRANS.H
2   *              Microsoft TPC-C Kit Ver. 3.00.000
3   *              Audited 08/23/96    By Francois Raab
4   *  PURPOSE:    Header file for ISAPI TPCC.DLL,
defines structures and functions used in the isapi
tpcc.dll.
5   *
6   *              Copyright Microsoft inc. 1996, All
Rights Reserved
7   *
8   *  Author:     PhilipDu, from tpcc.h by DamienL
9   *              DamienL@Microsoft.com
10  *              philipdu@Microsoft.com
11  */
12
13 #ifndef _INC_TRANS
14
15     #define _INC_TRANS
16
17     #ifdef USE_ODBC
18         #ifndef TIMESTAMP_STRUCT
19             #include <sqltypes.h>
20             #include <sql.h>
21             #include <sqlext.h>
```

```
        #endif
    #else
        #ifndef _INC_SQLFRONT
            #define DBNTWIN32
            #include <sqlfront.h>
            #include <sqldb.h>
        #endif
    #endif

    #ifndef DBINT
        typedef long DBINT;
    #endif

    #define DEFCLPACKSIZE               4096
    #define DEADLOCKWAIT                10

    // 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
    #define I_DATA_LEN                  50
    #define I_NAME_LEN                  24
    #define BRAND_LEN                    1
    #define LAST_NAME_LEN               16
    #define W_NAME_LEN                  10
    #define ADDRESS_LEN                 20
    #define STATE_LEN                    2
    #define ZIP_LEN                      9
    #define S_DIST_LEN                  24
    #define S_DATA_LEN                  50
    #define D_NAME_LEN                  10
    #define FIRST_NAME_LEN              16
    #define MIDDLE_NAME_LEN              2
    #define PHONE_LEN                   16
    #define DATETIME_LEN                30
    #define CREDIT_LEN                   2
    #define C_DATA_LEN                 250
    #define H_DATA_LEN                  24
    #define DIST_INFO_LEN               24
    #define MAX_OL_NEW_ORDER_ITEMS      15
    #define MAX_OL_ORDER_STATUS_ITEMS   15
    #define STATUS_LEN                  25
    #define OL_DIST_INFO_LEN            24

    // transaction structures

    typedef struct
    {
        short               ol_supply_w_id;
        long                ol_i_id;
        char                ol_i_name[I_NAME_LEN+1];
        short               ol_quantity;
        char
ol_brand_generic[BRAND_LEN+1];
        double              ol_i_price;
        double              ol_amount;
        short               ol_stock;
        short               num_warehouses;
    } OL_NEW_ORDER_DATA;

    typedef struct
    {
        short               w_id;
        short               d_id;
        long                c_id;
        short               o_ol_cnt;
        char                c_last[LAST_NAME_LEN+1];
        char                c_credit[CREDIT_LEN+1];
        double              c_discount;
        double              w_tax;
        double              d_tax;
        long                o_id;
        short               o_commit_flag;
    #ifdef USE_ODBC
        TIMESTAMP_STRUCT    o_entry_d;
    #else
        DBDATEREC           o_entry_d;
    #endif
        short               o_all_local;
        double              total_amount;
        long                num_deadlocks;
        char
execution_status[STATUS_LEN];
        OL_NEW_ORDER_DATA Ol[MAX_OL_NEW_ORDER_ITEMS];
    } 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;
    #ifdef USE_ODBC
        TIMESTAMP_STRUCT    h_date;
    #else
        DBDATEREC           h_date;
    #endif
        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];
        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];
        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];
    #ifdef USE_ODBC
        TIMESTAMP_STRUCT     c_since;
    #else
        DBDATEREC            c_since;
    #endif
        char                 c_credit[CREDIT_LEN+1];
        double               c_credit_lim;
        double               c_discount;
        double               c_balance;
        char                 c_data[200+1];
        long                 num_deadlocks;
        char
execution_status[STATUS_LEN];
    } PAYMENT_DATA;

    typedef struct
    {
        long                ol_i_id;
        short               ol_supply_w_id;
        short               ol_quantity;
        double              ol_amount;
    #ifdef USE_ODBC
        TIMESTAMP_STRUCT    ol_delivery_d;
    #else
        DBDATEREC           ol_delivery_d;
    #endif
    } OL_ORDER_STATUS_DATA;

    typedef struct
    {
        short                w_id;
        short                d_id;
        long                 c_id;
```

```
 172         char               c_first[FIRST_NAME_LEN+1];
 173          char
c_middle[MIDDLE_NAME_LEN+1];
 174          char                c_last[LAST_NAME_LEN+1];
 175          double              c_balance;
 176          long                o_id;
 177      #ifdef USE_ODBC
 178          TIMESTAMP_STRUCT    o_entry_d;
 179      #else
 180          DBDATEREC           o_entry_d;
 181      #endif
 182          short               o_carrier_id;
 183          OL_ORDER_STATUS_DATA OlOrderStatus-
Data[MAX_OL_ORDER_STATUS_ITEMS];
 184          short               o_ol_cnt;
 185          long                num_deadlocks;
 186          char
execution_status[STATUS_LEN];
 187      } ORDER_STATUS_DATA;
 188
 189      typedef struct
 190      {
 191          long                o_id;
 192      } DEL_ITEM;
 193
 194      typedef struct
 195      {
 196          short               w_id;
 197          short               o_carrier_id;
 198          SYSTEMTIME          queue_time;
 199          long                num_deadlocks;
 200          DEL_ITEM            DelItems[10];
 201          char
execution_status[STATUS_LEN];
 202      } DELIVERY_DATA;
 203
 204      typedef struct
 205      {
 206          short               w_id;
 207          short               d_id;
 208          short               thresh_hold;
 209          long                low_stock;
 210          long                num_deadlocks;
 211          char
execution_status[STATUS_LEN];
 212      } STOCK_LEVEL_DATA;
 213
 214  #endif
```

tux.h

```
   1  #ifndef TUX_H_INCLUDED
   2  #define TUX_H_INCLUDED
   3
   4  #define SERVICE_CHARS 32
   5
   6  #define PERF_MEASURE
   7
   8  #if defined(_DEBUG) && !defined(PERF_MEASURE)
   9  #define PERF_MEASURE
  10  #endif
  11
  12  typedef union
  13  {
  14      NEW_ORDER_DATA      NewOrderData;
  15      PAYMENT_DATA        PaymentData;
  16      ORDER_STATUS_DATA   OrderStatusData;
  17      DELIVERY_DATA       DeliveryData;
  18      STOCK_LEVEL_DATA    StockLevelData;
  19      char            ErrorMsg[4000]; // ack!!
  20  } TRANS_DATA;
  21
  22  typedef struct
  23  {
  24      int TermId;
  25      int SyncId;
  26      int bDeadlock;
  27      int bFailed;
  28      short DeadlockRetry;
  29      int Error;
  30      int Return;
  31  #ifdef PERF_MEASURE
  32     LARGE_INTEGER liTuxTime; // transaction time in ms
  33  #endif
  34  #ifdef TRANS_SEQ
  35      int Seq;
  36  #endif
  37      // Note: Trans must be last
  38      TRANS_DATA Trans;
  39  } TUX_DATA;
  40
  41  typedef struct
  42  {
  43      char Service[SERVICE_CHARS];
  44      // Note: Data must be last
  45      TUX_DATA Data;
  46  } TUX_MSG;
  47
  48  // macros to compute the size of various bits of
TUX_MSG.  It is
  49  // not enough to just add up the fields because of
possible alignment
  50  // issues
  51
  52  #define DATA_HEADER_SIZE(p) ((DWORD)(((char *)&(p)-
>Trans) - ((char *)(p))))
  53  #define MSG_HEADER_SIZE(p)  ((DWORD)(((char *)&(p)-
>Data.Trans) - ((char *)(p))))
  54
  55  #define NEW_ORDER_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(NEW_ORDER_DATA))
  56  #define PAYMENT_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(PAYMENT_DATA))
  57  #define ORDER_STATUS_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(ORDER_STATUS_DATA))
  58  #define DELIVERY_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(DELIVERY_DATA))
  59  #define STOCK_LEVEL_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(STOCK_LEVEL_DATA))
  60
  61
  62  #endif
```

tux_client.c

```
   1  #include <windows.h>
   2  #include <stdio.h>
   3  #include <string.h>
   4  #include <direct.h>
   5  #ifdef _DEBUG
   6  #include <time.h>
   7  #endif
   8
   9  #include "atmi.h"              /* TUXEDO  Header File
*/
  10
  11  #ifdef USE_ODBC
  12          #include <sqltypes.h>
  13          #include <sql.h>
  14          #include <sqlext.h>
  15          HENV    henv;
  16  #else
  17          #define DBNTWIN32
  18          #include <sqlfront.h>
  19          #include <sqldb.h>
  20  #endif
  21
  22  #include "trans.h"
  23  #include "tpcc.h"
  24  #include "pipe_routines.h"
  25  #include "util.h"
  26
  27  #include "tux.h"
  28
```

```
29  #define SERVICE_BUF_SIZE 16
30
31  typedef char * EXTENSION_CONROL_BLOCK;
32
33  const int TIMEOUT=1000*30;   // timeout in millisec-
onds
34  const int ARGSIZE=1024;
35  const char
*LOG_FILE="c:\\temp\\tpcc_logs\\tux_client\\client_%d.tx
t";
36
37  // Global variables set as parameters
38  int InitialCreate=0;
39  int ClientNumber=0;
40
41  char *TuxBuffer;
42
43  BOOL TuxInit()
44  {
45      BOOL bReturn = FALSE;
46
47      if (tpinit((TPINIT *) NULL) == -1)
48          fprintf(stderr, "tpinit failed\n");
49      else
50      {
51          TuxBuffer = (char *) tpalloc("CARRAY", NULL,
sizeof(TUX_MSG));
52
53          if (TuxBuffer != NULL)
54              bReturn = TRUE;
55          else
56          {
57              fprintf(stderr, "tpalloc of buffer
failed\n");
58              tpterm();
59          }
60      }
61
62      return bReturn;
63  }
64
65  void TuxCleanup(void)
66  {
67      tpfree(TuxBuffer);
68      tpterm();
69  }
70
71  BOOL TuxTransaction(char *Service, void *Data, long
BufSize, long *pnRead)
72  {
73
74      memcpy(TuxBuffer, Data, BufSize);
75
76  #ifdef _DEBUG
77      Log("about to tpcall Service %s, bufsize=%d\n",
Service, BufSize);
78  #endif
79      if (tpcall(Service, TuxBuffer, BufSize, &Tux-
Buffer, pnRead, TPNOTIME) == -1)
80      {
81          fprintf(stderr, "TuxTransaction: tpcall
failed, tperrno=%d\n", tperrno);
82          return FALSE;
83      }
84
85  #ifdef _DEBUG
86      Log("tp call retuned %d bytes\n", *pnRead);
87  #endif
88
89      if (*pnRead < BufSize)
90      {
91          fprintf(stderr, "TuxTransaction: nRead(%d) <
BufSize(%d)\n", *pnRead, BufSize);
92          return FALSE;
93      }
94
95      memcpy(Data, TuxBuffer, *pnRead);
96      return TRUE;
97  }
98
99  void
100 HandleTransactions(HANDLE hPipe)
101 {
102     TUX_MSG msg;
103     DWORD nRead;
104     HANDLE hEvent;
105
106     hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
107     if (hEvent == INVALID_HANDLE_VALUE)
108     {
109         fprintf(stderr, "Unable to create event han-
dle\n");
110         return;
111     }
112
113     while(ReadPipe(hPipe, hEvent, &msg, sizeof(msg),
&nRead))
114     {
115         DWORD nWritten;
116
117         if (!TuxTransaction(msg.Service, &msg.Data,
sizeof(msg.Data), &nRead))
118         {
119             fprintf(stderr, "TuxTransaction
failed\n");
120             break;
121         }
122         if (!WritePipe(hPipe, hEvent, &msg, nRead,
&nWritten))
123         {
124             fprintf(stderr,"WritePipe Failed in Han-
dleTransactions()\n");
125             break;
126         }
127         if (nWritten != nRead)
128         {
129             fprintf(stderr, "HandleTransactions:
nWritten(%d) != nRead(%d)\n",
130                 nWritten, nRead);
131         }
132     }
133
134     CloseHandle(hEvent);
135 }
136
137 BOOL
138 StartAnother(char *name, int number, int InitialCre-
ate)
139 {
140     STARTUPINFO si;
141     PROCESS_INFORMATION pi;
142     char args[1024];
143
144     sprintf(args, "%s -n %d %d", name, number, Ini-
tialCreate);
145
146     memset(&si, 0, sizeof(si) );
147     si.cb = sizeof(si);
148
149     // Start the child process.
150     if( !CreateProcess( NULL, // No module name (use
command line).
151         args,          // Command line.
152         NULL,            // Process handle not inher-
itable.
153         NULL,            // Thread handle not inher-
itable.
154         FALSE,          // Set handle inheritance to
FALSE.
155         0,               // No creation flags.
156         NULL,           // Use parent's environment
block.
157         NULL,            // Use parent's starting
directory.
```

```
158          &si,              // Pointer to STARTUPINFO
 structure.
159          &pi )             // Pointer to
 PROCESS_INFORMATION structure.
160      )
161      {
162          fprintf(stderr, "Unable to start another, num-
 ber=%d\n",number);
163      return FALSE;
164      }
165
166      return TRUE;
167  }
168
169  void
170  Usage(char *ProgName)
171  {
172      fprintf(stderr,"usage: %s <initial create>\n",
 ProgName);
173  }
174
175  BOOL
176  Parse(int argc, char **argv)
177  {
178      int c;
179      BOOL bReturn=TRUE;
180      extern char *optarg;
181      extern int optind, optopt;
182
183      while(bReturn && ((c = getopt(argc, argv, "n:"))
 != -1 ))
184      {
185          switch(c)
186          {
187              case 'n':
188                  ClientNumber = atoi(optarg);
189                  if (ClientNumber <=0)
190                      bReturn = FALSE;
191                  break;
192              case ':':
193                fprintf(stderr, "option %c requires an
 argument\n", optopt);
194                  bReturn = FALSE;
195                  break;
196              case '?':
197                  bReturn = FALSE;
198                  break;
199              default:
200                  // should not happen
201                  fprintf(stderr, "Parse in default
 case.\n");
202                  bReturn = FALSE;
203                  break;
204          }
205      }
206
207      // See if we have any arguments left
208      switch (argc - optind)
209      {
210          case 1:
211              InitialCreate = atoi(argv[optind]);
212              if (InitialCreate < 0)
213              {
214                  bReturn = FALSE;
215                  break;
216              }
217              // fall through
218          case 0:
219              // nothing else specified - OK
220              break;
221          default:
222             fprintf(stderr,"only one <initial_create>
 allowed\n");
223              bReturn = FALSE;
224              break;
225      }
226
227      return bReturn;
228  }
229
230  void
231  SetUpStderr(void)
232  {
233      char buf[_MAX_PATH];
234
235      if (!mkpath(LOG_FILE))
236      {
237          fprintf(stderr, "mkpath failed for %s\n",
 LOG_FILE);
238          exit(1);
239      }
240
241      sprintf(buf, LOG_FILE, ClientNumber);
242      freopen(buf, "w", stderr);
243      setbuf(stderr, NULL);
244  }
245
246  int
247  main(int argc, char **argv)
248  {
249      HANDLE hPipe;
250
251      if (!Parse(argc, argv))
252      {
253          Usage(argv[0]);
254          exit(1);
255      }
256
257  #ifdef _DEBUG
258      Log("client %d starting (as thread 0x%x)\n", Cli-
 entNumber, GetCurrentThreadId());
259  #endif
260
261      SetUpStderr();
262
263      if (!TuxInit())
264      {
265          fprintf(stderr, "tuxinit failed\n");
266          exit(1);
267      }
268
269      if (ClientNumber == 0)
270      {
271          int i;
272  #ifdef _DEBUG
273          Log("Doing initial create of %d\n", Initial-
 Create);
274  #endif
275          for (i=1;i<InitialCreate;i++)
276              StartAnother(argv[0], i, InitialCreate);
277      }
278
279      hPipe = OpenServerPipe(ClientNumber, INFINITE);
280
281      if (hPipe == INVALID_HANDLE_VALUE)
282          fprintf(stderr, "OpenServerPipe failed,
 error=%d\n", GetLastError());
283      else
284      {
285
286          if (ClientNumber >= InitialCreate-1)
287              StartAnother(argv[0], ClientNumber + 1,
 InitialCreate);
288
289          HandleTransactions(hPipe);
290
291          CloseHandle(hPipe);
292      }
293
294      TuxCleanup();
295
296      return 0;
297  }
```

tux_server.c

```
#include <windows.h>
#include <stdio.h>
#include <time.h>
#include <stdarg.h>

// Tuxedo include files
#include <atmi.h>
#include <userlog.h>

// Database include files

#ifdef USE_ODBC
        #include <sqltypes.h>
        #include <sql.h>
        #include <sqlext.h>
        HENV    henv;
#else
        #define DBNTWIN32
        #include <sqlfront.h>
        #include <sqldb.h>
#endif

// include files for this project

#include "util.h"
#include "trans.h"
#include "tpcc.h"
#include "sqlroutines.h"
#include "tux.h"

// Global variables
short iMaxConnections=1;
char szErrorLog-
Path[]="c:\\temp\\tpcc_logs\\tux_server.txt";
DBPROCESS *pdbproc;
char *Server=NULL;;
char *Database="tpcc";
char *User="sa";
char *Password="";
int spId;
TUX_DATA data;
TERM Term;
EXTENSION_CONTROL_BLOCK *gpECB=NULL;

#ifdef PERF_MEASURE
LARGE_INTEGER CounterFrequencyMS;
#endif

void TuxLog(char *format, ...)
{
    va_list args;
    char buf[4096];
    int len;

    va_start(args, format);
    _strtime(buf);
    strcat(buf," ");
    len = strlen(buf);
    (void)_vsnprintf(buf+len, sizeof(buf)-len-1, for-
mat, args);
    buf[sizeof(buf)-1]='\0';
    va_end(args);

    userlog(buf);
}

void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char
*szStr)
{
    strcpy(data.Trans.ErrorMsg, szStr);
    data.Error = 1;
}

BOOL IsValidTermId(int TermId)
{
    return FALSE;
}

int
tpsvrinit(int argc, char *argv[])
{
    char App[1024];
#ifdef PERF_MEASURE
    LARGE_INTEGER liFrequency;
#endif

    mkpath(szErrorLogPath);

    TuxLog("starting the tuxedo TPCC server");

    if (gethostname(App, sizeof(App)))
        strcpy(App, "TPCC");

    if (!SQLInit())
    {
        TuxLog("SQLInit failed");
        return -1;
    }

    if (getenv("SERVER"))
        Server = strdup(getenv("SERVER"));

    if (Server == NULL)
    {
        TuxLog("SERVER Environment variable not
 set");
        return -1;
    }

    if (SQLOpenConnection(NULL, 0, 0, &pdbproc,
 Server, Database, User, Password, App, &spId))
    {
        TuxLog("SQLOpenconnection failed");
        SQLCleanup();
        return -1;
    }

#ifdef PERF_MEASURE
    QueryPerformanceFrequency(&liFrequency);
    CounterFrequencyMS.QuadPart = liFrequency.Quad-
 Part/1000; // in ms
#endif

    return 0;
}

void
tpsvrdone(void)
{
    TuxLog("shuttting down the tuxedo TPCC server");
    free(Server);
    SQLCloseConnection(NULL, pdbproc);
    SQLCleanup();
}

void
NEW_ORDER(TPSVCINFO *rqst)
{
    PECBINFO pECBInfo = SQLGetECB(pdbproc);
    int size = rqst->len;

#ifdef PERF_MEASURE
    LARGE_INTEGER liStartCounter, liEndCounter;
    QueryPerformanceCounter(&liStartCounter);
#endif
#ifdef TRANS_SEQ
    TuxLog("B 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
#endif
#ifdef _DEBUG
    if (((TUX_DATA*)rqst->data)->Trans.NewOrder-
 Data.c_id)
        TuxLog("Start NEW_ORDER for customer %d,
```

```
 size=%d data=0x%x\n",((TUX_DATA*)rqst->data)-
 >Trans.NewOrderData.c_id, size, rqst->data);
146      else
147          TuxLog("Start NEW_ORDER for customer %s
 size=%d data=0x%x\n",((TUX_DATA*)rqst->data)-
 >Trans.NewOrderData.c_last, size, rqst->data);
148  #endif
149
150      memcpy(&data, rqst->data, size);
151
152      data.Return = SQLNewOrder(NULL,  data.TermId,
 data.SyncId, pdbproc, &data.Trans.NewOrderData,
 data.DeadlockRetry);
153
154      data.bDeadlock = pECBInfo->bDeadlock;
155      data.bFailed = pECBInfo->bFailed;
156
157      if (data.Error)
158      {
159          size = sizeof(data);
160          strcpy(data.Trans.ErrorMsg, Term.pClient-
 Data[0].szBuffer);
161      }
162
163  #ifdef PERF_MEASURE
164      QueryPerformanceCounter(&liEndCounter);
165      data.liTuxTime.QuadPart = (liEndCounter.QuadPart
 - liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart;
166  #endif
167
168  #ifdef _DEBUG
169      TuxLog("End NEWORDER, in %I64dms bFailed=%d\n",
 data.liTuxTime, data.bFailed);
170  #endif
171
172      memcpy(rqst->data, &data, size);
173  #ifdef TRANS_SEQ
174      TuxLog("E 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
175  #endif
176      tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
177  }
178
179  void
180  STOCK_LEVEL(TPSVCINFO *rqst)
181  {
182      PECBINFO pECBInfo = SQLGetECB(pdbproc);
183      int size = rqst->len;
184  #ifdef PERF_MEASURE
185      LARGE_INTEGER liStartCounter, liEndCounter;
186      QueryPerformanceCounter(&liStartCounter);
187  #endif
188  #ifdef TRANS_SEQ
189      TuxLog("B 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
190  #endif
191  #ifdef _DEBUG
192      TuxLog("Start STOCK_LEVEL data=0x%x\n", rqst-
 >data);
193  #endif
194
195      memcpy(&data, rqst->data, size);
196
197      data.Return = SQLStockLevel(NULL,  data.TermId,
 data.SyncId, pdbproc, &data.Trans.StockLevelData,
 data.DeadlockRetry);
198      data.bDeadlock = pECBInfo->bDeadlock;
199      data.bFailed = pECBInfo->bFailed;
200
201      if (data.Error)
202      {
203          size = sizeof(data);
204          strcpy(data.Trans.ErrorMsg, Term.pClient-
 Data[0].szBuffer);
205      }
206
207  #ifdef PERF_MEASURE
208      QueryPerformanceCounter(&liEndCounter);
209      data.liTuxTime.QuadPart = (liEndCounter.QuadPart
 - liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart;
210  #endif
211
212  #ifdef _DEBUG
213      TuxLog("End STOCK_LEVEL, in %I64dms
 bFailed=%d\n", data.liTuxTime, data.bFailed);
214  #endif
215
216      memcpy(rqst->data, &data, size);
217  #ifdef TRANS_SEQ
218      TuxLog("E 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
219  #endif
220      tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
221  }
222
223  void
224  PAYMENT(TPSVCINFO *rqst)
225  {
226      PECBINFO pECBInfo = SQLGetECB(pdbproc);
227      int size = rqst->len;
228
229  #ifdef PERF_MEASURE
230      LARGE_INTEGER liStartCounter, liEndCounter;
231      QueryPerformanceCounter(&liStartCounter);
232  #endif
233  #ifdef TRANS_SEQ
234      TuxLog("B 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
235  #endif
236  #ifdef _DEBUG
237      if (((TUX_DATA*)rqst->data)->Trans.Payment-
 Data.c_id)
238          TuxLog("Start PAYMENT for customer %d
 data=0x%x\n",((TUX_DATA*)rqst->data)->Trans.Payment-
 Data.c_id, rqst->data);
239      else
240          TuxLog("Start PAYMENT for customer %s
 data=0x%x\n",((TUX_DATA*)rqst->data)->Trans.Payment-
 Data.c_last, rqst->data);
241  #endif
242
243      memcpy(&data, rqst->data, size);
244
245      data.Return = SQLPayment(NULL,  data.TermId,
 data.SyncId, pdbproc, &data.Trans.PaymentData,
 data.DeadlockRetry);
246
247      data.bDeadlock = pECBInfo->bDeadlock;
248      data.bFailed = pECBInfo->bFailed;
249
250      if (data.Error)
251      {
252          size = sizeof(data);
253          strcpy(data.Trans.ErrorMsg, Term.pClient-
 Data[0].szBuffer);
254      }
255
256  #ifdef PERF_MEASURE
257      QueryPerformanceCounter(&liEndCounter);
258      data.liTuxTime.QuadPart = (liEndCounter.QuadPart
 - liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart;
259  #endif
260
261  #ifdef _DEBUG
262      TuxLog("End PAYMENT, in %I64dms bFailed=%d\n",
 data.liTuxTime,  data.bFailed);
263  #endif
264      memcpy(rqst->data, &data, size);
265  #ifdef TRANS_SEQ
266      TuxLog("E 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
267  #endif
268      tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
269  }
270
271  void
272  ORDER_STATUS(TPSVCINFO *rqst)
273  {
274      PECBINFO pECBInfo = SQLGetECB(pdbproc);
```

```
275      int size = rqst->len;
276
277  #ifdef PERF_MEASURE
278      LARGE_INTEGER liStartCounter, liEndCounter;
279      QueryPerformanceCounter(&liStartCounter);
280  #endif
281  #ifdef TRANS_SEQ
282      TuxLog("B 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
283  #endif
284  #ifdef _DEBUG
285      TuxLog("Start ORDER_STATUS data=0x%x\n", rqst-
  >data);
286  #endif
287
288      memcpy(&data, rqst->data, size);
289
290      data.Return = SQLOrderStatus(NULL,  data.TermId,
  data.SyncId, pdbproc, &data.Trans.OrderStatusData,
  data.DeadlockRetry);
291      data.bDeadlock = pECBInfo->bDeadlock;
292      data.bFailed = pECBInfo->bFailed;
293
294      if (data.Error)
295      {
296          size = sizeof(data);
297          strcpy(data.Trans.ErrorMsg, Term.pClient-
  Data[0].szBuffer);
298      }
299
300  #ifdef PERF_MEASURE
301      QueryPerformanceCounter(&liEndCounter);
302      data.liTuxTime.QuadPart = (liEndCounter.QuadPart
  - liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart;
303  #endif
304
305  #ifdef _DEBUG
306      TuxLog("End ORDER_STATUS, in %I64dms
  bFailed=%d\n", data.liTuxTime, data.bFailed);
307  #endif
308
309      memcpy(rqst->data, &data, size);
310  #ifdef TRANS_SEQ
311      TuxLog("E 0x%x\n", ((TUX_DATA*)rqst->data)->Seq);
312  #endif
313      tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
314  }
```

tux_server.mak

```
1  !IF "$(CFG)" == ""
2  CFG=Debug
3  !MESSAGE No configuration specified.  Defaulting to
Debug
4  !ENDIF
5
6  !IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
7  !MESSAGE Invalid configuration "$(CFG)" specified.
8  !MESSAGE You can specify a configuration when running
NMAKE on this makefile
9  !MESSAGE by defining the macro CFG on the command
line.  For example:
10 !MESSAGE
11 !MESSAGE NMAKE CFG="Debug"
12 !MESSAGE
13 !MESSAGE Possible choices for configuration are:
14 !MESSAGE
15 !MESSAGE "Release"
16 !MESSAGE "Debug"
17 !MESSAGE
18 !ERROR An invalid configuration is specified.
19 !ENDIF
20
21 OUT_PATH    = c:\temp\mckee\objs\tpcc\tux_server
22 SRCDIR          = ..\Src
23 OBJDIR          = $(OUT_PATH)\$(CFG)
24 OUTDIR          = $(OUT_PATH)\Bin
25 TUX     = c:\tux_dev
26 SQL     = ..\sqlptk
27
28 TUXINCLUDE  = $(TUX)\include
29 SQLINCLUDE  = $(SQL)\include
30
31 !IF "$(CFG)" != "Debug"
32 LDEBUG       =
33 CDEBUG       =
34 LDEBUG_RG    =
35 CDEBUG_RG    =
36 DEBUG         =
37 FLAGS         = /D "WIN32" /D "_WINDOWS"
38 OPT           = /Ot
39 !ELSE
40 LDEBUG       = /debug /pdb:$(OBJDIR)\tux_server.pdb
41 CDEBUG       = /Zi /Yd
42 FLAGS         = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
43 OPT           = /Od
44 !ENDIF
45
46 OBJS    = "$(OBJDIR)\tux_server.obj" "$(OBJDIR)\sql-
routines.obj" "$(OBJDIR)\error.obj" "$(OBJDIR)\util.obj"
47 FLAGS   = /D "TUX" /c /nologo /MD /W3 $(FLAGS) $(CDE-
BUG) $(OPT) /I$(SQLINCLUDE) /Fd$(OBJDIR)\tux_server.pdb
48
49 ALL:    $(OBJDIR)\. $(OBJDIR)\tux_server.exe
50
51 $(OBJDIR)\.:
52     if not exist $(OBJDIR) md $(OBJDIR)
53
54 "$(OBJDIR)\tux_server.obj":
55     cl.exe $(FLAGS) /I$(TUXINCLUDE)
/Fd$(OBJDIR)\tux_server.pdb /Fo$(OBJDIR)\tux_server.obj
/c "$(SRCDIR)\tux_server.c"
56
57 "$(OBJDIR)\sqlroutines.obj":
58     cl.exe $(FLAGS) /Fo$(OBJDIR)\sqlroutines.obj
"$(SRCDIR)\sqlroutines.c"
59
60 "$(OBJDIR)\error.obj":
61     cl.exe $(FLAGS) /Fo$(OBJDIR)\error.obj
"$(SRCDIR)\error.c"
62
63 "$(OBJDIR)\util.obj":   "$(SRCDIR)\util.c"
"$(SRCDIR)\util.h"
64     cl.exe $(FLAGS) /Fo$(OBJDIR)\util.obj
"$(SRCDIR)\util.c"
65
66 $(OBJDIR)\tux_server.exe:   $(OBJS)
67     build.cmd "$(OBJDIR)"
```

tux_sql.c

```
1  #include <windows.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <time.h>
5
6  #ifdef USE_ODBC
7          #include <sqltypes.h>
8          #include <sql.h>
9          #include <sqlext.h>
10         HENV    henv;
11 #else
12         #define DBNTWIN32
13         #include <sqlfront.h>
14         #include <sqldb.h>
15 #endif
16
17 #include "trans.h"
18 #include "httpext.h"
19 #include "tpcc.h"
20 #include "tux.h"
21 #include "sqlroutines.h"
22 #include "pipe_routines.h"
23 #include "util.h"
24 #include "tux_trans.h"
```

```
25
26
27
28  BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK
 *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc,
 char *server, char *database, char *user, char *pass-
 word, char *app, int *spid)
29  {
30      PECBINFO    pEcbInfo;
31
32      //set pECB data into dbproc
33      pEcbInfo = (PECBINFO)malloc(sizeof(ECBINFO));
34
35      pEcbInfo->bDeadlock = FALSE;
36      pEcbInfo->pECB      = pECB;
37      pEcbInfo->iTermId   = iTermId;
38      pEcbInfo->iSyncId   = iSyncId;
39
40      *dbproc = (DBPROCESS *)pEcbInfo;
41
42      return FALSE;
43  }
44
45  BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK
 *pECB, DBPROCESS *dbproc)
46  {
47      return FALSE;
48  }
49
50  BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int
 iTermId, int iSyncId, DBPROCESS *dbproc,
 STOCK_LEVEL_DATA *pStockLevel, short deadlock_retry)
51  {
52      long ReceiveLen = sizeof(STOCK_LEVEL_DATA);
53
54      return TuxTransaction("STOCK_LEVEL", pECB, iTer-
 mId, iSyncId, dbproc, deadlock_retry, pStockLevel,
 sizeof(*pStockLevel));
55
56  }
57
58  int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int
 iTermId, int iSyncId, DBPROCESS *dbproc,
 NEW_ORDER_DATA  *pNewOrder, short deadlock_retry)
59  {
60      return TuxTransaction("NEW_ORDER", pECB, iTermId,
 iSyncId, dbproc, deadlock_retry, pNewOrder, sizeof(*pNe-
 wOrder));
61  }
62
63  int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int
 iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA
 *pPayment, short deadlock_retry)
64  {
65      return TuxTransaction("PAYMENT", pECB, iTermId,
 iSyncId, dbproc, deadlock_retry, pPayment, sizeof(*pPay-
 ment));
66  }
67
68  int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int
 iTermId, int iSyncId, DBPROCESS *dbproc,
 ORDER_STATUS_DATA *pOrderStatus, short deadlock_retry)
69  {
70      return TuxTransaction("ORDER_STATUS", pECB, iTer-
 mId, iSyncId, dbproc, deadlock_retry, pOrderStatus,
 sizeof(*pOrderStatus));
71  }
72
73  PECBINFO SQLGetECB(PDBPROCESS p)
74  {
75      return (PECBINFO)p;
76  }
```

tux_trans.c

```
 1  #include <windows.h>
 2  #include <stdio.h>
 3  #include <string.h>
 4  #include <time.h>
 5
 6  #include <atmi.h>
 7
 8  #ifdef USE_ODBC
 9          #include <sqltypes.h>
10          #include <sql.h>
11          #include <sqlext.h>
12          HENV    henv;
13  #else
14          #define DBNTWIN32
15          #include <sqlfront.h>
16          #include <sqldb.h>
17  #endif
18
19  #include "trans.h"
20  #include "httpext.h"
21  #include "tpcc.h"
22  #include "tux.h"
23  #include "sqlroutines.h"
24  #include "pipe_routines.h"
25  #include "util.h"
26  #include "tux_trans.h"
27
28  static  CRITICAL_SECTION        CriticalSection;
29
30  void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char
*szStr);
31
32  volatile int ThreadCount;
33  volatile DWORD TlsIndex=0xffffffff;
34
35  TPINIT * volatile pTPInit=NULL;
36
37  #ifdef PERF_MEASURE
38  LARGE_INTEGER liCounterFrequencyMS;
39
40  typedef enum eTRANS {NEW_ORDER, ORDER_STATUS,
STOCK_LEVEL, PAYMENT, DELIVERY, TRANS_TYPE_COUNT} TRANS;
41
42  typedef struct t_TPCC_PERF_COUNTER
43  {
44      ULONG ulGenerationStarted;
45      CRITICAL_SECTION cs;
46      __int64 iCount;
47      __int64 iSUTTime;
48      __int64 iTUXTime;
49      ULONG ulGenerationDone;
50  } TPCC_PERF_COUNTER;
51
52  TPCC_PERF_COUNTER *Counters=NULL;
53  char *CounterNames[] = {"New Order", "Order Status",
"Stock Level", "Payment", "Delivery", NULL};
54
55  #define PERF_OUTPUT
56  #ifdef PERF_OUTPUT
57  volatile ULONG ulCounter=0;
58  #endif
59  #endif
60
61
62  /*
63   * This file contains the tuxedo client side routines.
Basically, they
64   * are stubs for the routines in sqlroutines.c, which
are used by the
65   * tuxedo server
66   */
67
68  BOOL DoInit(int ClientNumber)
69  {
70
71  #ifdef _DEBUG
72      fprintf(stderr, "DoInit begins for thread %d,
pTPInit=0x%x\n", ThreadCount, pTPInit);
```

```
 73 #endif
 74
 75     if (!pTPInit)
 76     {
 77         pTPInit = (TPINIT *)tpalloc("TPINIT", NULL,
sizeof(TPINIT));
 78
 79         if (!pTPInit)
 80         {
 81            fprintf(stderr, "tpalloc of pTPInit failed
for thread %d\n",
 82                  ThreadCount);
 83             return FALSE;
 84         }
 85
 86         pTPInit->flags |= TPMULTICONTEXTS;
 87     }
 88
 89     _snprintf(pTPInit->cltname, sizeof(*pTPInit->clt-
name), "cl%d", ClientNumber);
 90
 91     if (tpinit(pTPInit) == -1)
 92     {
 93         fprintf(stderr, "tpinit failed for thread %d,
tperrno=%d\n",
 94              ThreadCount, tperrno);
 95         return FALSE;
 96     }
 97
 98 #ifdef _DEBUG
 99     fprintf(stderr, "tpinit succeeded for thread
 %d\n", ThreadCount);
100 #endif
101     TlsSetValue(TlsIndex, (void *)TRUE);
102
103     return TRUE;
104 }
105
106 BOOL SQLThreadAttach(void)
107 {
108     BOOL bReturn = TRUE;
109
110 #ifdef _DEBUG
111     fprintf(stderr, "thread %d about to attach\n",
 GetCurrentThreadId());
112 #endif
113     EnterCriticalSection(&CriticalSection);
114
115     if (!DoInit(ThreadCount))
116     {
117         Log("thread %d unable to Initialize in SQLTh-
 readAttach()\n", GetCurrentThreadId());
118         bReturn=FALSE;
119     }
120
121     if (bReturn)
122         ThreadCount++;
123
124     LeaveCriticalSection(&CriticalSection);
125 #ifdef _DEBUG
126     fprintf(stderr, "thread %d attach returns %d\n",
 GetCurrentThreadId(), bReturn);
127 #endif
128
129     return bReturn;
130 }
131
132 BOOL SQLThreadDetach(void)
133 {
134
135 #ifdef _DEBUG
136     fprintf(stderr, "thread %d detaching\n", GetCur-
 rentThreadId());
137 #endif
138     EnterCriticalSection(&CriticalSection);
139     ThreadCount--;
140     LeaveCriticalSection(&CriticalSection);
141
142     tpterm();
143
144     return TRUE;
145 }
146
147 BOOL SQLInit(void)
148 {
149     // Perform one time initialization.
150 #ifdef PERF_MEASURE
151     int i;
152
153     LARGE_INTEGER liFrequency;
154     QueryPerformanceFrequency(&liFrequency);
155     liCounterFrequencyMS.QuadPart = liFrequency.Quad-
 Part/1000;
156
157     Counters = (TPCC_PERF_COUNTER *)cal-
 loc(TRANS_TYPE_COUNT, sizeof(*Counters));
158
159     if (Counters == NULL)
160     {
161         Log("calloc of Counters failed\n");
162         return FALSE;
163     }
164
165     for(i=0;i<TRANS_TYPE_COUNT;i++)
166         InitializeCriticalSection(&Counters[i].cs);
167
168 #endif
169
170 #ifdef _DEBUG
171     fprintf(stderr, "SQLInit() called\n");
172 #endif
173     InitializeCriticalSection(&CriticalSection);
174
175     TlsIndex = TlsAlloc();
176
177     if (TlsIndex == 0xffffffff)
178     {
179         Log("TlsAlloc Failed\n");
180         return FALSE;
181     }
182
183     return SQLThreadAttach();
184 }
185
186 void SQLCleanup(void)
187 {
188 #ifdef _DEBUG
189     fprintf(stderr, "SQLCleanup() called\n");
190 #endif
191 #ifdef PERF_MEASURE
192     {
193         int i;
194         for(i=0;i<TRANS_TYPE_COUNT;i++)
195             DeleteCriticalSection(&Counters[i].cs);
196     }
197 #endif
198     SQLThreadDetach();
199     tpfree((char *)pTPInit);
200     pTPInit = NULL;
201     DeleteCriticalSection(&CriticalSection);
202 }
203
204 static int
205 ExceptionFilter(int iException)
206 {
207     Log("thread %d caught exception 0x%x\n", GetCur-
 rentThreadId(), iException);
208
209     return EXCEPTION_EXECUTE_HANDLER;
210 }
211
212 void
213 LogTransData(char *Service, TUX_DATA *pData)
214 {
```

```
215     if (!strcmp(Service, "PAYMENT"))
216     {
217         if (pData->Trans.PaymentData.c_id)
218            Log("Thread %d Payment for Customer=%d\n",
219                 GetCurrentThreadId(),
220                 pData->Trans.PaymentData.c_id);
221         else
222            Log("Thread %d Payment for Customer=%s\n",
223                 GetCurrentThreadId(),
224                 pData->Trans.PaymentData.c_last);
225     }
226     else if (!strcmp(Service, "NEW_ORDER"))
227     {
228         if (pData->Trans.NewOrderData.c_id)
229             Log("Thread %d NewOrder for Cus-
 tomer=%d\n",
230                 GetCurrentThreadId(),
231                 pData->Trans.NewOrderData.c_id);
232         else
233             Log("Thread %d NewOrder for Cus-
 tomer=%s\n",
234                 GetCurrentThreadId(),
235                 pData->Trans.NewOrderData.c_last);
236     }
237     else
238     {
239         Log("Thread %d %s\n", GetCurrentThreadId(),
 Service);
240     }
241 }
242
243 int TuxTransaction(char *Service,
 EXTENSION_CONTROL_BLOCK   *pECB,
244     int TermId, int SyncId, DBPROCESS *dbproc, short
 DeadlockRetry, void *Data, long BufSize)
245 {
246     TUX_DATA *pData;
247     DWORD nRead;
248     PECBINFO pECBInfo = (PECBINFO)dbproc; // forgive
 them them, for they know not what they do...
249     int iReturn;
250 #ifdef PERF_MEASURE
251     LARGE_INTEGER liStartCounter, liEndCounter;
252     TPCC_PERF_COUNTER *pCounter;
253     __int64 iSutms, iTotalms;
254 #endif
255
256     // we are pessimistic here
257     pECBInfo->bFailed = TRUE;
258
259     if (!TlsGetValue(TlsIndex))
260     {
261         if (!SQLThreadAttach())
262         {
263             Log("thread %d attach failed\n", GetCur-
 rentThreadId());
264             return -1;
265         }
266     }
267
268     pData = (TUX_DATA *) tpalloc("CARRAY", NULL,
 sizeof(TUX_DATA));
269
270     if (!pData)
271     {
272         Log("thread %d tpalloc() failed\n", GetCur-
 rentThreadId());
273         return -1;
274     }
275
276 #ifdef _DEBUG
277     Log("thread %d TuxTransaction pData=0x%x\n", Get-
 CurrentThreadId(), pData);
278     Log("thread %d BufSize=%d, sizeof(TUX_DATA)=%d
 sizeof(TRANS_DATA)=%d\n", GetCurrentThreadId(), Buf-
 Size, sizeof(TUX_DATA), sizeof(TRANS_DATA));
279 #endif
280
281     // fill the struct to ship to tux
282     memcpy(&pData->Trans, Data, BufSize);
283     pData->TermId = TermId;
284     pData->SyncId = SyncId;
285     pData->DeadlockRetry = DeadlockRetry;
286     pData->Error = FALSE;
287 #ifdef TRANS_SEQ
288     {
289         char *p = strstr(pECB->lpszQueryString,
 "SEQ=");
290
291         if (p)
292             pData->Seq = strtoul(p+4, NULL, 0);
293     }
294 #endif
295     BufSize += DATA_HEADER_SIZE(pData); // Send the
 headers too
296
297 #ifdef _DEBUG
298     Log("thread %d about to tpcall Service %s, buf-
 size=%d\n", GetCurrentThreadId(), Service, BufSize);
299
300     LogTransData(Service, pData);
301 #endif
302
303 #ifdef PERF_MEASURE
304     QueryPerformanceCounter(&liStartCounter);
305 #endif
306
307     __try
308     {
309 #ifdef TRANS_SEQ
310         Log("C 0x%x\n", pData->Seq);
311 #endif
312         if (tpcall(Service, (char *)pData, BufSize,
 &(char *)pData, &nRead, 0) == -1)
313         {
314             Log("thread %d tpcall failed tper-
 rno=%d\n", GetCurrentThreadId(), tperrno);
315             tpfree((char *)pData);
316
317             return -1;
318         }
319 #ifdef TRANS_SEQ
320         Log("R 0x%x\n", pData->Seq);
321 #endif
322     }
323     __except(ExceptionFilter(GetExceptionCode()))
324     {
325         Log("thread %d handling an exception from
 tpcall.  Service=%s, pData=0x%x, BufSize=%d\n", GetCur-
 rentThreadId(), Service, pData, BufSize);
326         // we nest the try here, since it is fairly
 likely that
327         // we got here because pData was bad in the
 tpcall.
328         __try
329         {
330             LogTransData(Service, pData);
331         }
332         __except(ExceptionFilter(GetExceptionCode()))
333         {
334             Log("thread %d LogTransData() caused an
 exception\n", GetCurrentThreadId());
335         }
336         tpfree((char *)pData);
337         return -1;
338     }
339
340 #ifdef PERF_MEASURE
341     QueryPerformanceCounter(&liEndCounter);
342     iTotalms = (liEndCounter.QuadPart - liStart-
 Counter.QuadPart)/liCounterFrequencyMS.QuadPart;
343     iSutms = pData->liTuxTime.QuadPart;
344
345     switch (*Service)
```

```
346     {
347         case 'N':
348             pCounter = Counters + NEW_ORDER;
349             break;
350         case 'P':
351             pCounter = Counters +  PAYMENT;
352             break;
353         case 'O':
354             pCounter = Counters +  ORDER_STATUS;
355             break;
356         case 'S':
357             pCounter = Counters +  STOCK_LEVEL;
358             break;
359         default:
360            Log("Unable to determine Trans for %s\n",
 Service);
361             break;
362     }
363
364     EnterCriticalSection(&pCounter->cs);
365     InterlockedIncrement(&pCounter->ulGeneration-
 Started);
366
367     pCounter->iCount++;
368     pCounter->iSUTTime += iSutms;
369     pCounter->iTUXTime += iTotalms - iSutms;
370
371     InterlockedIncrement(&pCounter->ulGeneration-
 Done);
372     LeaveCriticalSection(&pCounter->cs);
373
374 #ifdef PERF_OUTPUT
375     if ((InterlockedIncrement((ULONG *)&ulCounter) %
 5000) == 0)
376     {
377         int i;
378
379         Log("Counters after %lu transactions\n",
 ulCounter);
380
381         for(i=0;i<TRANS_TYPE_COUNT;i++)
382         {
383             pCounter = Counters + i;
384             EnterCriticalSection(&pCounter->cs);
385             if (pCounter->iCount)
386                 Log("%s %I64d %I64d (%I64d) %I64d
 (%I64d)\n", CounterNames[i],
387                     pCounter->iCount,
388                   pCounter->iSUTTime, pCounter->iSUT-
 Time/pCounter->iCount,
389                     pCounter->iTUXTime, pCounter->iTUX-
 Time/pCounter->iCount);
390             LeaveCriticalSection(&pCounter->cs);
391         }
392     }
393 #endif
394
395 #ifdef DEBUG
396     Log("thread %d end %s in %I64dms (tux=%I64dms,
 sut=%I64dms), %d bytes pData=0x%x, bFailed=%d\n", GetCur-
 rentThreadId(), Service,
397             iTotalms, iTotalms - iSutms, iSutms,
 nRead, pData, pData->bFailed);
398 #endif
399 #endif
400
401     if (nRead < (DWORD)BufSize)
402     {
403         Log("thread %d nRead(%d) < BufSize(%d)\n",
 GetCurrentThreadId(), nRead, BufSize);
404         tpfree((char *)pData);
405         return -1;
406     }
407     else if (nRead > (DWORD)BufSize)
408     {
409 #ifdef _DEBUG
410         Log("thread %d nRead(%d) > BufSize(%d)\n",
 GetCurrentThreadId(), nRead, BufSize);
411 #endif
412     }
413
414     BufSize -= DATA_HEADER_SIZE(pData); // Ignore the
 headers now
415
416     if (pData->Error)
417     {
418 #ifdef _DEBUG
419         Log("thread %d pData->Error set,
 ErrorMsg=%s\n", GetCurrentThreadId(), pData-
 >Trans.ErrorMsg);
420 #endif
421         WriteZString(pECB, pData->Trans.ErrorMsg);
422     }
423
424     // patch things up so the upper levels don't know
 this went
425     // through tux
426
427     pECBInfo->iTermId = TermId;
428     pECBInfo->iSyncId = SyncId;
429     pECBInfo->bDeadlock = pData->bDeadlock;
430     pECBInfo->bFailed = pData->bFailed;
431
432
433     if (pData->Error)
434         Log("thread %d, Term=%d pData->Error set,
 ErrorMsg=%s\n", GetCurrentThreadId(), TermId, pData-
 >Trans.ErrorMsg);
435     if (pData->bFailed)
436         Log("thread %d Term=%d pData->bFailed is
 True\n", GetCurrentThreadId(), TermId);
437
438 #ifdef _DEBUG
439     // it is Ok to get Return == 0, because some
 transactions are supposed
440     // to fail.  Only note if for _DEBUG
441     if (!pData->Return)
442        Log("Thread %d pData->Return == FALSE\n", Get-
 CurrentThreadId());
443 #endif
444
445     memcpy(Data, &pData->Trans, BufSize);
446
447     iReturn = pData->Return;
448
449     tpfree((char *)pData);
450
451     return iReturn;
452 }
```

tux_trans.h

```
1  #ifndef __TUX_INIT_H_INCLUDED
2  #define __TUX_INIT_H_INCLUDED
3
4  int TuxTransaction(char *Service,
EXTENSION_CONTROL_BLOCK   *pECB,
5      int TermId, int SyncId, DBPROCESS *dbproc, short
DeadlockRetry,
6      void *Data, long BufSize);
7
8  #endif
```

tux_trans_client.c

```
1  #include <windows.h>
2  #include <stdio.h>
3
4
5  #ifdef USE_ODBC
6          #include <sqltypes.h>
7          #include <sql.h>
8          #include <sqlext.h>
```

```
 9          HENV    henv;
10  #else
11          #define DBNTWIN32
12          #include <sqlfront.h>
13          #include <sqldb.h>
14  #endif
15
16  #include "trans.h"
17  #include "httpext.h"
18  #include "tpcc.h"
19  #include "tux.h"
20  #include "sqlroutines.h"
21  #include "pipe_routines.h"
22  #include "util.h"
23  #include "tux_trans.h"
24
25  const int ARGSIZE=1024;
26  const int PIPE_BUF_SIZE=4096;
27
28  static  CRITICAL_SECTION        CriticalSection;
29
30  void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char
*szStr);
31
32  typedef struct
33  {
34      int ThreadNumber;
35      HANDLE hPipe;
36  } THREAD_DATA;
37
38  /*
39   * This file contains the tuxedo client side rou-
tines.  Basically, they
40   * are stubs for the routines in sqlroutines.c, which
are used by the
41   * tuxedo server
42   */
43
44  DWORD TlsIndex;
45  int  ThreadCount=0;
46
47  BOOL SQLThreadAttach(void)
48  {
49      THREAD_DATA *pData;
50
51  #ifdef _DEBUG
52      fprintf(stderr, "SQLThread attach starts\n");
53  #endif
54      pData = (THREAD_DATA *)mal-
loc(sizeof(THREAD_DATA));
55
56      if (!pData)
57          return FALSE;
58
59      memset(pData, 0, sizeof(*pData));
60
61      EnterCriticalSection(&CriticalSection);
62      pData->ThreadNumber = ThreadCount++;
63      LeaveCriticalSection(&CriticalSection);
64
65      pData->hPipe = OpenClientPipe(pData->ThreadNum-
ber);
66      if (pData->hPipe == INVALID_HANDLE_VALUE)
67      {
68          fprintf(stderr, "SQLThreadattach failed for
thread %d\n", pData->ThreadNumber);
69          free(pData);
70          return FALSE;
71      }
72      else
73          TlsSetValue(TlsIndex, pData);
74
75  #ifdef _DEBUG
76      fprintf(stderr, "SQLThread attach succeeds for
thread %d\n", pData->ThreadNumber);
77  #endif
78      return TRUE;
79  }
80
81  BOOL SQLThreadDetach(void)
82  {
83      THREAD_DATA *pData = TlsGetValue(TlsIndex);
84
85      if (pData)
86      {
87          CloseHandle(pData->hPipe);
88          free(pData);
89      }
90
91      return TRUE;
92  }
93
94  BOOL SQLInit(void)
95  {
96      // Perform one time initialization.  According to
the comments in tpcc.c, this will
97      // be called once when the DLL is loaded.  We
assume that is true, and also that
98      // the caller has protected the call with a crit-
ical section.
99
100     InitializeCriticalSection(&CriticalSection);
101
102     TlsIndex = TlsAlloc();
103
104     if (TlsIndex == 0xffffffff)
105     {
106         MessageBox(NULL, "TlsAlloc failed", "Init",
 MB_OK | MB_ICONSTOP);
107         return FALSE;
108     }
109 #ifdef _DEBUG
110     fprintf(stderr, "TlsIndex = %d\n", TlsIndex);
111 #endif
112 }
113
114 void SQLCleanup(void)
115 {
116     TlsFree(TlsIndex);
117     TlsIndex = 0xffffffff;
118     DeleteCriticalSection(&CriticalSection);
119 }
120
121 BOOL TuxTransaction(char *Service,
 EXTENSION_CONTROL_BLOCK  *pECB,
122     int TermId, int SyncId, DBPROCESS *dbproc, short
 DeadlockRetry, void *Data, long BufSize)
123 {
124     THREAD_DATA *pData;
125     TUX_MSG msg;
126     DWORD nBytes;
127     PECBINFO pECBInfo = (PECBINFO)dbproc; // forgive
 them them, for they know not what they do...
128
129     // we are pessimistic here
130     pECBInfo->bFailed = TRUE;
131
132     pData = TlsGetValue(TlsIndex);
133     if (pData == NULL)
134     {
135         if (!SQLThreadAttach())
136         {
137           fprintf(stderr, "TuxTransaction: unable to
 attach\n");
138             return FALSE;
139         }
140         pData = TlsGetValue(TlsIndex);
141     }
142
143     // fill the struct to ship to tux
144     strcpy(msg.Service, Service);
145     msg.Data.TermId = TermId;
146     msg.Data.SyncId = SyncId;
147     msg.Data.DeadlockRetry = DeadlockRetry;
```

```
148      msg.Data.Error = FALSE;
149      memcpy(&msg.Data.Trans, Data, BufSize);
150
151      if (!WritePipe(pData->hPipe, NULL, &msg,
  MSG_HEADER_SIZE(&msg)+BufSize, &nBytes))
152      {
153          char error_buf[1024];
154          ErrorString(error_buf, sizeof(error_buf),
  GetLastError());
155          fprintf(stderr, "Tuxtransaction: WritePipe
  Failed [%s]\n", error_buf);
156          return FALSE;
157      }
158
159      if (nBytes != MSG_HEADER_SIZE(&msg)+BufSize)
160      {
161         fprintf(stderr, "Tuxtransaction: short write,
  size=%d, written=%d\n",
162              MSG_HEADER_SIZE(&msg)+BufSize, nBytes);
163          return FALSE;
164      }
165
166      if (!ReadPipe(pData->hPipe, NULL, &msg,
  sizeof(msg), &nBytes))
167      {
168          char error_buf[1024];
169          ErrorString(error_buf, sizeof(error_buf),
  GetLastError());
170          fprintf(stderr, "Tuxtransaction: ReadPipe
  Failed [%s]\n", error_buf);
171          return FALSE;
172      }
173
174      if (msg.Data.Error)
175      {
176  #ifdef _DEBUG
177          fprintf(stderr, "msg.Error set,
  ErrorMsg=%s\n", msg.Data.Trans.ErrorMsg);
178  #endif
179
180          WriteZString(pECB, msg.Data.Trans.ErrorMsg);
181      }
182
183      // patch things up so the upper levels don't know
  this went
184      // through tux
185
186      pECBInfo->iTermId = TermId;
187      pECBInfo->iSyncId = SyncId;
188      pECBInfo->bDeadlock = msg.Data.bDeadlock;
189      pECBInfo->bFailed = msg.Data.bFailed;
190
191  #ifdef _DEBUG
192      if (msg.Data.Error)
193          fprintf(stderr, "Term=%d msg.Error set,
  ErrorMsg=%s\n", TermId, msg.Data.Trans.ErrorMsg);
194      if (msg.Data.bFailed)
195          fprintf(stderr, "Term=%d msg.Data.bFailed is
  True\n", TermId);
196
197  #endif
198      memcpy(Data, &msg.Data.Trans, BufSize);
199      return msg.Data.Return;
200  }
```

tuxedo_process.mak

```
1   !IF "$(CFG)" == ""
2   CFG=Debug
3   !MESSAGE No configuration specified.  Defaulting to
Debug
4   !ENDIF
5
6   !IF "$(SQL_LOC)" == ""
7   SQL_LOC=..\..\sqlptk
8   !MESSAGE No SQL_LOC specified.  Defaulting to
$(SQL_LOC)
9   !ENDIF
10
11  !IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
12  !MESSAGE Invalid configuration "$(CFG)" specified.
13  !MESSAGE You can specify a configuration when running
NMAKE on this makefile
14  !MESSAGE by defining the macro CFG on the command
line.  For example:
15  !MESSAGE
16  !MESSAGE NMAKE CFG="Debug"
17  !MESSAGE
18  !MESSAGE Possible choices for configuration are:
19  !MESSAGE
20  !MESSAGE "Release"
21  !MESSAGE "Debug"
22  !MESSAGE
23  !ERROR An invalid configuration is specified.
24  !ENDIF
25
26  OUT_PATH    =
c:\temp\mckee\objs\tpcc\dll\tuxedo_process
27  SRCDIR          = ..\..\Src
28  OBJDIR          = $(OUT_PATH)\$(CFG)
29  OUTDIR          = $(OUT_PATH)\Bin
30  ODBC        = \progra~1\msdev
31
32  DBLIB           = $(SQL_LOC)
33  DBLIBINC        = $(DBLIB)\INCLUDE
34  ODBCINCDIR  = $(ODBC)\INCLUDE
35  DBLIBDIR        = $(DBLIB)\LIB
36  ODBCLIBDIR  = $(ODBC)\LIB
37
38  !IF "$(CFG)" != "Debug"
39  LDEBUG       =
40  CDEBUG       =
41  LDEBUG_RG    =
42  CDEBUG_RG    =
43  DEBUG        =
44  FLAGS        = /D "WIN32" /D "_WINDOWS"
45  OPT          = /Ot
46  !ELSE
47  LDEBUG       = /debug /pdb:$(OBJDIR)\tpcc1.pdb
48  CDEBUG       = /Zi /Yd
49  LDEBUG_RG    = /debug /pdb:$(OBJDIR)\install.pdb
50  CDEBUG_RG    = /Zi /Yd /Fd$(OBJDIR)\install.pdb
51  FLAGS        = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
52  OPT          = /Od
53  !ENDIF
54
55  LINK32_LIBS1    = user32.lib msacm32.lib advapi32.lib
$(DBLIBDIR)\ntwdblib.lib
56  LINK32_OBJS1    = "$(OBJDIR)\tpcc1.obj"
"$(OBJDIR)\tpcc1.res" "$(OBJDIR)\tux_sql.obj"
"$(OBJDIR)\tux_trans_client.obj" "$(OBJDIR)\error.obj"
"$(OBJDIR)\util.obj" "$(OBJDIR)\pipe_routines.obj"
57  LINK32_DEF1     = "$(SRCDIR)\tpcc1.def"
58  LINK32_FLAGS1   = /nologo /subsystem:windows /dll
/incremental:no $(LDEBUG) /def:"$(LINK32_DEF1)"
/out:"$(OBJDIR)\tpcc.dll"
59
60  LINK32_LIBS2    = user32.lib msacm32.lib advapi32.lib
$(ODBCLIBDIR)\odbc32.LIB
61  LINK32_OBJS2    = "$(OBJDIR)\tpcc2.obj"
"$(OBJDIR)\tpcc2.res"
62  LINK32_DEF2     = "$(SRCDIR)\tpcc2.def"
63  LINK32_FLAGS2   = /nologo /subsystem:windows /dll
/incremental:no $(LDEBUG) /def:"$(LINK32_DEF2)"
/out:"$(OBJDIR)\tpcc2.dll"
64
65  LINK32_LIBS_RG  = user32.lib gdi32.lib advapi32.lib
version.lib comctl32.lib
66  LINK32_OBJS_RG  = "$(OBJDIR)\install.obj"
"$(OBJDIR)\install.res"
67  LINK32_FLAGS_RG = /nologo /subsystem:windows /incre-
mental:no $(LDEBUG_RG) /out:$(OUTDIR)\install.exe
68
69  ALL:    $(OBJDIR)\. $(OBJDIR)\tpcc.dll
```

```
70
71  $(OBJDIR)\.:
72      if not exist $(OBJDIR) md $(OBJDIR)
73
74  $(OUTDIR)\.:
75      if not exist $(OUTDIR) md $(OUTDIR)
76
77  "$(OBJDIR)\tpcc1.obj":   "$(SRCDIR)\tpcc.c"
 "$(SRCDIR)\tpcc.h"
78      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
 /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
 /Fo$(OBJDIR)\tpcc1.obj /c "$(SRCDIR)\tpcc.c"
79
80  "$(OBJDIR)\tux_trans_client.obj":
 "$(SRCDIR)\tux_trans_client.c" "$(SRCDIR)\tpcc.h"
81      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
 /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
 /Fo$(OBJDIR)\tux_trans_client.obj /c
 "$(SRCDIR)\tux_trans_client.c"
82
83  "$(OBJDIR)\tux_sql.obj":   "$(SRCDIR)\tux_sql.c"
 "$(SRCDIR)\tpcc.h"
84      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
 /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
 /Fo$(OBJDIR)\tux_sql.obj /c "$(SRCDIR)\tux_sql.c"
85
86  "$(OBJDIR)\pipe_routines.obj":
 "$(SRCDIR)\pipe_routines.c" "$(SRCDIR)\tpcc.h"
87      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
 /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
 /Fo$(OBJDIR)\pipe_routines.obj /c
 "$(SRCDIR)\pipe_routines.c"
88
89  "$(OBJDIR)\error.obj":   "$(SRCDIR)\error.c"
 "$(SRCDIR)\error.h"
90      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
 /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
 /Fo$(OBJDIR)\error.obj /c "$(SRCDIR)\error.c"
91
92  "$(OBJDIR)\util.obj":   "$(SRCDIR)\util.c"
 "$(SRCDIR)\util.h"
93      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
 /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
 /Fo$(OBJDIR)\util.obj /c "$(SRCDIR)\util.c"
94
95  $(OBJDIR)\tpcc1.res:     $(SRCDIR)\tpcc1.rc
96      rc.exe /l 0x409 /fo $(OBJDIR)\tpcc1.res $(FLAGS)
 $(SRCDIR)\tpcc1.rc
97
98  $(OBJDIR)\tpcc.dll:     $(LINK32_OBJS1)
 $(LINK32_DEF1)
99      link.exe   $(LINK32_FLAGS1) $(LINK32_OBJS1)
 $(LINK32_LIBS1)
100
101  "$(OBJDIR)\tpcc2.obj":   "$(SRCDIR)\tpcc.c"
  "$(SRCDIR)\tpcc.h"
102      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
  /I$(ODBCINCDIR) $(FLAGS) /Fd$(OBJDIR)\tpcc2.pdb
  /Fo$(OBJDIR)\tpcc2.obj /c /D"USE_ODBC"
  "$(SRCDIR)\tpcc.c"
103
104  $(OBJDIR)\tpcc2.res:     $(SRCDIR)\tpcc2.rc
105      rc.exe /l 0x409 /fo $(OBJDIR)\tpcc2.res $(FLAGS)
  $(SRCDIR)\tpcc2.rc
106
107  $(OBJDIR)\tpcc2.dll:     $(LINK32_OBJS2)
  $(LINK32_DEF2)
108      link.exe   $(LINK32_FLAGS2) $(LINK32_OBJS2)
  $(LINK32_LIBS2)
109
110  $(OBJDIR)\delisrv1.exe:  $(SRCDIR)\delisrv.c
  $(SRCDIR)\delisrv.h
111      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I
  $(DBLIBINC) $(FLAGS) /Fo$(OBJDIR)\delisrv.obj
  $(SRCDIR)\delisrv.c /link /out:$(OBJDIR)\delisrv1.exe
  $(DBLIBDIR)\ntwdblib.lib msacm32.lib advapi32.lib
112
113  $(OBJDIR)\delisrv2.exe:  $(SRCDIR)\delisrv.c
  $(SRCDIR)\delisrv.h
114      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
  /I$(ODBCINCDIR) $(FLAGS) /Fo$(OBJDIR)\delisrv.obj
  $(SRCDIR)\delisrv.c /D"USE_ODBC"  /link
  /out:$(OBJDIR)\delisrv2.exe $(ODBCLIBDIR)\odbc32.lib
  msacm32.lib advapi32.lib
115
116  $(OBJDIR)\install.res:  $(SRCDIR)\install.rc
  $(OBJDIR)\tpcc.dll $(OBJDIR)\tpcc2.dll
  $(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
117      rc.exe /l 0x409 /fo$(OBJDIR)\install.res /i
  $(OBJDIR) /i $(SRCDIR) $(FLAGS) $(SRCDIR)\install.rc
118
119  $(OBJDIR)\install.obj:  $(SRCDIR)\install.c
  $(OBJDIR)\tpcc.dll $(OBJDIR)\tpcc2.dll
  $(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
  $(OBJDIR)\install.res
120      cl -W3 $(CDEBUG_RG) /Fo$(OBJDIR)\install.obj /c
  $(SRCDIR)\install.c
121
122  $(OUTDIR)\install.exe:     $(OBJDIR)\install.obj
  $(OBJDIR)\install.res
123      link.exe @<<
124    $(LINK32_FLAGS_RG) $(LINK32_OBJS_RG)
  $(LINK32_LIBS_RG)
125  <<
```

tuxedo_threads.mak

```
 1  !IF "$(CFG)" == ""
 2  CFG=Debug
 3  !MESSAGE No configuration specified.  Defaulting to
Debug
 4  !ENDIF
 5
 6  !IF "$(SQL_LOC)" == ""
 7  SQL_LOC=..\..\sqlptk
 8  !MESSAGE No SQL_LOC specified.  Defaulting to
$(SQL_LOC)
 9  !ENDIF
10
11  !IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
12  !MESSAGE Invalid configuration "$(CFG)" specified.
13  !MESSAGE You can specify a configuration when running
NMAKE on this makefile
14  !MESSAGE by defining the macro CFG on the command
line.  For example:
15  !MESSAGE
16  !MESSAGE NMAKE CFG="Debug"
17  !MESSAGE
18  !MESSAGE Possible choices for configuration are:
19  !MESSAGE
20  !MESSAGE "Release"
21  !MESSAGE "Debug"
22  !MESSAGE
23  !ERROR An invalid configuration is specified.
24  !ENDIF
25
26  OUT_PATH    =
c:\temp\mckee\objs\tpcc\dll\tuxedo_threads
27  SRCDIR          = ..\..\Src
28  OBJDIR          = $(OUT_PATH)\$(CFG)
29  OUTDIR          = $(OUT_PATH)\Bin
30  ODBC        = \progra~1\msdev
31
32  DBLIB           = $(SQL_LOC)
33  DBLIBINC        = $(DBLIB)\INCLUDE
34  ODBCINCDIR  = $(ODBC)\INCLUDE
35  DBLIBDIR        = $(DBLIB)\LIB
36  ODBCLIBDIR  = $(ODBC)\LIB
37
38  TUX     = c:\tux_dev
39  TUXINCLUDE  = $(TUX)\INCLUDE
40  TUXLIB      = $(TUX)\LIB
41  TUXLIBS     = libtux.lib libbuft.lib libtux2.lib
libfml.lib libfml32.lib libgp.lib
```

```
42
43  !IF "$(CFG)" != "Debug"
44  LDEBUG       =
45  CDEBUG       =
46  LDEBUG_RG    =
47  CDEBUG_RG    =
48  DEBUG        =
49  FLAGS        = /D "WIN32" /D "_WINDOWS"
50  OPT          = /Ot
51  !ELSE
52  LDEBUG       = /debug /pdb:$(OBJDIR)\tpcc1.pdb
53  CDEBUG       = /Zi /Yd
54  LDEBUG_RG    = /debug /pdb:$(OBJDIR)\install.pdb
55  CDEBUG_RG    = /Zi /Yd /Fd$(OBJDIR)\install.pdb
56  FLAGS        = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
57  OPT          = /Od
58  !ENDIF
59
60  LINK32_LIBS1   = user32.lib msacm32.lib advapi32.lib
$(DBLIBDIR)\ntwdblib.lib $(TUXLIBS)
61  LINK32_OBJS1    = "$(OBJDIR)\tpcc1.obj"
"$(OBJDIR)\tpcc1.res" "$(OBJDIR)\tux_sql.obj"
"$(OBJDIR)\tux_trans.obj" "$(OBJDIR)\error.obj"
"$(OBJDIR)\util.obj" "$(OBJDIR)\pipe_routines.obj"
62  LINK32_DEF1     = "$(SRCDIR)\tpcc1.def"
63  LINK32_FLAGS1   = /nologo /subsystem:windows /dll
/incremental:no $(LDEBUG) /def:"$(LINK32_DEF1)"
/out:"$(OBJDIR)\tpcc.dll" /libpath:$(TUXLIB)
64
65  LINK32_LIBS2   = user32.lib msacm32.lib advapi32.lib
$(ODBCLIBDIR)\odbc32.LIB
66  LINK32_OBJS2    = "$(OBJDIR)\tpcc2.obj"
"$(OBJDIR)\tpcc2.res"
67  LINK32_DEF2     = "$(SRCDIR)\tpcc2.def"
68  LINK32_FLAGS2   = /nologo /subsystem:windows /dll
/incremental:no $(LDEBUG) /def:"$(LINK32_DEF2)"
/out:"$(OBJDIR)\tpcc2.dll"
69
70  LINK32_LIBS_RG  = user32.lib gdi32.lib advapi32.lib
version.lib comctl32.lib
71  LINK32_OBJS_RG  = "$(OBJDIR)\install.obj"
"$(OBJDIR)\install.res"
72  LINK32_FLAGS_RG = /nologo /subsystem:windows /incre-
mental:no $(LDEBUG_RG) /out:$(OUTDIR)\install.exe
73
74  ALL:    $(OBJDIR)\. $(OBJDIR)\tpcc.dll
75
76  $(OBJDIR)\.:
77      if not exist $(OBJDIR) md $(OBJDIR)
78
79  $(OUTDIR)\.:
80      if not exist $(OUTDIR) md $(OUTDIR)
81
82  "$(OBJDIR)\tpcc1.obj":   "$(SRCDIR)\tpcc.c"
"$(SRCDIR)\tpcc.h"
83      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
/Fo$(OBJDIR)\tpcc1.obj /c "$(SRCDIR)\tpcc.c"
84
85  "$(OBJDIR)\tux_trans.obj":   "$(SRCDIR)\tux_trans.c"
"$(SRCDIR)\tpcc.h"
86      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(TUX-
INCLUDE) /I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
/Fo$(OBJDIR)\tux_trans.obj /c "$(SRCDIR)\tux_trans.c"
87
88  "$(OBJDIR)\tux_sql.obj":   "$(SRCDIR)\tux_sql.c"
"$(SRCDIR)\tpcc.h"
89      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
/Fo$(OBJDIR)\tux_sql.obj /c "$(SRCDIR)\tux_sql.c"
90
91  "$(OBJDIR)\pipe_routines.obj":
"$(SRCDIR)\pipe_routines.c" "$(SRCDIR)\tpcc.h"
92      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
/Fo$(OBJDIR)\pipe_routines.obj /c
"$(SRCDIR)\pipe_routines.c"
93
94  "$(OBJDIR)\error.obj":   "$(SRCDIR)\error.c"
"$(SRCDIR)\error.h"
95      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
/Fo$(OBJDIR)\error.obj /c "$(SRCDIR)\error.c"
96
97  "$(OBJDIR)\util.obj":   "$(SRCDIR)\util.c"
"$(SRCDIR)\util.h"
98      cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(DBLIBINC) $(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb
/Fo$(OBJDIR)\util.obj /c "$(SRCDIR)\util.c"
99
100 $(OBJDIR)\tpcc1.res:     $(SRCDIR)\tpcc1.rc
101     rc.exe /l 0x409 /fo $(OBJDIR)\tpcc1.res $(FLAGS)
$(SRCDIR)\tpcc1.rc
102
103 $(OBJDIR)\tpcc.dll:     $(LINK32_OBJS1)
$(LINK32_DEF1)
104     link.exe   $(LINK32_FLAGS1) $(LINK32_OBJS1)
$(LINK32_LIBS1)
105
106 "$(OBJDIR)\tpcc2.obj":   "$(SRCDIR)\tpcc.c"
"$(SRCDIR)\tpcc.h"
107     cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(ODBCINCDIR) $(FLAGS) /Fd$(OBJDIR)\tpcc2.pdb
/Fo$(OBJDIR)\tpcc2.obj /c /D"USE_ODBC"
"$(SRCDIR)\tpcc.c"
108
109 $(OBJDIR)\tpcc2.res:     $(SRCDIR)\tpcc2.rc
110     rc.exe /l 0x409 /fo $(OBJDIR)\tpcc2.res $(FLAGS)
$(SRCDIR)\tpcc2.rc
111
112 $(OBJDIR)\tpcc2.dll:     $(LINK32_OBJS2)
$(LINK32_DEF2)
113     link.exe   $(LINK32_FLAGS2) $(LINK32_OBJS2)
$(LINK32_LIBS2)
114
115 $(OBJDIR)\delisrv1.exe:  $(SRCDIR)\delisrv.c
$(SRCDIR)\delisrv.h
116     cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I
$(DBLIBINC) $(FLAGS) /Fo$(OBJDIR)\delisrv.obj
$(SRCDIR)\delisrv.c /link /out:$(OBJDIR)\delisrv1.exe
$(DBLIBDIR)\ntwdblib.lib msacm32.lib advapi32.lib
117
118 $(OBJDIR)\delisrv2.exe:  $(SRCDIR)\delisrv.c
$(SRCDIR)\delisrv.h
119     cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT)
/I$(ODBCINCDIR) $(FLAGS) /Fo$(OBJDIR)\delisrv.obj
$(SRCDIR)\delisrv.c /D"USE_ODBC"  /link
/out:$(OBJDIR)\delisrv2.exe $(ODBCLIBDIR)\odbc32.lib
msacm32.lib advapi32.lib
120
121 $(OBJDIR)\install.res:  $(SRCDIR)\install.rc
$(OBJDIR)\tpcc.dll $(OBJDIR)\tpcc2.dll
$(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
122     rc.exe /l 0x409 /fo$(OBJDIR)\install.res /i
$(OBJDIR) /i $(SRCDIR) $(FLAGS) $(SRCDIR)\install.rc
123
124 $(OBJDIR)\install.obj:  $(SRCDIR)\install.c
$(OBJDIR)\tpcc.dll $(OBJDIR)\tpcc2.dll
$(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
$(OBJDIR)\install.res
125     cl -W3 $(CDEBUG_RG) /Fo$(OBJDIR)\install.obj /c
$(SRCDIR)\install.c
126
127 $(OUTDIR)\install.exe:     $(OBJDIR)\install.obj
$(OBJDIR)\install.res
128     link.exe @<<
129   $(LINK32_FLAGS_RG) $(LINK32_OBJS_RG)
$(LINK32_LIBS_RG)
130 <<
```

update.cmd

```
1 set
DEST=\\15.81.94.207\c$\home\bretm\src\win32\tpcc\dll
```

```
net use \\15.81.94.207\c$ /user:tpcc
copy *.h %DEST%\src
copy *.c %DEST%\src
```

util.c

```
#include <windows.h>
#include <direct.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdarg.h>
#include <time.h>
#include "util.h"


/* FUNCTION: void UtilStrCpy(char * pDest, char * pSrc, int n)
 *
 * PURPOSE: This function copies n characters from string pSrc to pDst and places a
 *          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;
}

// ErrorString turns an error number returned by GetLastError() into
// a human readable string (kinda like perror)

void
ErrorString(char *buf, int buf_size, DWORD dwError)
{

    int nBytes;

    nBytes = FormatMessage(
        FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        dwError,
       MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
        buf,
        buf_size,
        NULL
    );

    if (nBytes>2)
    {
        nBytes -= 2;
        buf[nBytes] = '\0';
    }
    else
        _snprintf(buf, buf_size, "Unable to get error message for error %d", dwError);
}


// mkpath is like mkdir -p (that is, it makes all the components of the
// path, not just the last one.
BOOL
mkpath(const char *path)
{
    BOOL bReturn = TRUE;

    if (strlen(path))
    {
        const char *pos = path;

        // skip the drive letter if there is one.
        if (*(pos+1) == ':')
            pos += 2;

        while ((pos = strchr(pos+1, '\\')) != NULL)
        {
            char buf[MAX_PATH+1];
            extern int errno;

            strcpy(buf, path);
            buf[pos-path] = '\0';

            if (_mkdir(buf) < 0 && (errno != EEXIST))
            {
                bReturn = FALSE;
                break;
            }

        }
    }

    return bReturn;
}


void Log(char *format, ...)
{
    va_list args;
    char buf[4096];
    int len;

    va_start(args, format);
    _strtime(buf);
    strcat(buf," ");
    len = strlen(buf);
    (void)_vsnprintf(buf+len, sizeof(buf)-len-1, format, args);
    buf[sizeof(buf)-1]='\0';
    va_end(args);

    fputs(buf, stderr);
}
```

util.h

```
#ifndef TPCC_UTIL_H
#define TPCC_UTIL_H

void UtilStrCpy(char * pDest, char * pSrc, int n);
BOOL IsValidTermId(int TermId);
void ErrorString(char *buf, int buf_size, DWORD dwError);
BOOL mkpath(const char *path);
void Log(char *format, ...);
#endif
```

A.2 Driver

/driver/convert_iis_delilog.c

```
 1  #include <stdio.h>
 2  #include <string.h>
 3
 4  #include "tpcc.h"
 5
 6  #define MAX_LINE_SIZE 128
 7
 8
 9  void
10  parse_date(char *buf, struct tm *tm)
11  {
12      tm->tm_year = atoi(buf);
13      buf += 3;
14      tm->tm_mon  = atoi(buf) - 1;
15      buf += 3;
16      tm->tm_mday = atoi(buf);
17
18  }
19
20
21  /* returns the left over ms */
22  int
23  parse_time(char *buf, struct tm *tm)
24  {
25      tm->tm_hour = atoi(buf);
26      buf += 3;
27      tm->tm_min = atoi(buf);
28      buf += 3;
29      tm->tm_sec = atoi(buf);
30      buf += 3;
31
32      return atoi(buf);
33  }
34
35  int convert_deli_line(char *buf, delivery_trans *t)
36  {
37      char *pos;
38      int i;
39      struct tm tm;
40      time_t time;
41      int ms, elapsed;
42
43      /* if we get here at all, the status was OK */
44      t->status = OK;
45
46      /* we use strtok to parse the string.  if we have
too few
47       * parameters, we will notice in the loop below.
If we
48       * have too many we will notice after the loop
49       */
50
51      /* now figure out the time stuff */
52
53      memset(&tm, 0, sizeof(tm));
54      parse_date(strtok(buf, ","), &tm);
55      ms = parse_time(strtok(NULL, ","), &tm);
56     (void)strtok(NULL,","); /* throw away the end time
*/
57
58      time = mktime(&tm);
59
60      elapsed = atoi(strtok(NULL,","));
61
62      t->enque[0].tv_sec = time;
63      t->enque[0].tv_usec = ms * 1000;
64
65      t->deque[0] = t->enque[0]; /* we don't have this
one */
66
67      /* calculate completion time */
68      t->complete[0] = t->deque[0];       /* start with
deque time */
69      t->complete[0].tv_usec += elapsed * 1000;   /* add
elapsed time */
70
71      /* adjust so tv_usec < 1000000 */
72      t->complete[0].tv_sec += t->com-
plete[0].tv_usec/1000000;
73      t->complete[0].tv_usec %= 1000000;
74
75      t->W_ID = atoi(strtok(NULL,","));
76      t->O_CARRIER_ID = atoi(strtok(NULL,","));
77
78      for(i=0;i<10;i++)
79      {
80          char *pos;
81          pos = strtok(NULL, ",");
82          if (pos == NULL)
83          {
84              fprintf(stderr,"too few fields\n");
85              exit(1);
86          }
87          t->order[i].O_ID = atoi(pos);
88          t->order[i].status = OK;
89      }
90
91      if (strtok(NULL,",") != NULL)
92      {
93          fprintf(stderr,"wrong number of fields\n");
94          exit(1);
95      }
96  }
97
98  int
99  read_deli_line(FILE *fp, char *buf)
100 {
101     int i=0;
102
103     while(i < MAX_LINE_SIZE)
104     {
105         char ch = fgetc(fp);
106
107         if (feof(fp))
108             break;
109
110         if (ch == '\r')
111             if (i)
112                 break;
113             else
114                 continue;
115         if (ch != '\n')
116             buf[i++] = ch;
117     }
118
119     if (i == MAX_LINE_SIZE)
120     {
121         fprintf(stderr,"line too long\n");
122         exit(1);
123     }
124
125     buf[i] = '\0';
126
127     if (feof(fp) && i == 0)
128         return 0; /* end of file */
129
130     if (buf[0] == '*')
131         return 0; /* error line */
132
133     return 1;
134 }
135
136 int
137 main(int argc, char **argv)
138 {
139     char buf[MAX_LINE_SIZE];
140     FILE *infp = stdin,
141          *outfp = stdout;
142
143     while(read_deli_line(infp, buf))
144     {
145         delivery_trans t;
146         convert_deli_line(buf, &t);
147         fwrite(&t, sizeof(t), 1, outfp);
```

```
148     }
149
150     return 0;
151 }
```

/driver/driver.c

```
1
/*******************************************************
**********************
 2  @(#) Version: A.10.10 $Date: 97/02/27 23:27:04 $
 3
 4  (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*******************************************************
**********************/
 6 #include <stdio.h>
 7 #include <values.h>
 8 #include <unistd.h>
 9 #include <time.h>
10 #include <sys/types.h>
11 #include <sys/ipc.h>
12 #include <sys/shm.h>
13 #include <fcntl.h>
14 #include <signal.h>
15 #include <math.h>
16
17 #include "tpcc.h"
18 #include "shm.h"
19 #include "random.h"
20
21 #include <time.h>
22
23 #ifdef TKJOS_DEBUG
24 #define LOG(X) { \
25     FILE *x; \
26     x=fopen("/tmp/driver.log","a"); \
27     fprintf X ;\
28     fclose(x); }
29 #else
30 #define LOG(X)
31 #endif
32
33 #ifdef RAMP_UP
34 /* we do spawn_rate forks/spawn_period secods */
35 /* these are the defaults, and they can be set via
environment variables */
36 int spawn_rate=2;
37 int spawn_period=1;
38 #endif
39
40
41 void initialize_info();
42 extern int CLAST_CONST_C;
43 extern int CID_CONST_C;
44 extern int IID_CONST_C;
45
46 extern int trans_type;
47
48 int userid = -1;
49 int info_fd = -1;
50
51 /* Key time delays */
52 double key[] =     {0.,  18.0,   3.0,   2.0,   2.0,
2.0 };
53
54 /* Think time delays */
55 double think[] =   {0.,  12.5,  12.5,  10.4,   5.4,
5.4};
56
57 /** Emulex response time factors **/
58 /*double m_fudge[] = {0.,  .56,  .41,   .32,  .45,
.46,     .0};
59   double t_fudge[] = {0.,  .83,  .35,   .47,   .29,
.27,   .0}; */
60
61 /** no concentrator time factors **/
62 double m_fudge[] = {.0,  .0,   .0,    .0,    .0,    .0,
.0};
63 double t_fudge[] = {.0,  .0,   .0,    .0,    .0,    .0,
.0};
64
65 int main(argn, argv)
66
/*******************************************************
**********
67 TPC-C driver program
68
*******************************************************
*********/
69      int argn;
70      char **argv;
71      {
72
73      /* initialize everything */
74      initialize_stuff(argn, argv);
75
76     /* fill in the table of last Names for Payment and
OrderStatus */
77      GenerateLastNames();
78
79      /* repeat until told to stop and no more users
exist */
80      stopflag = NO;
81      while (!stopflag || attached > 0)
82      {
83      double delay_time=5.0;
84
85          if (attached != desired || post_transactions
== YES)
86          delay_time = 1.0;
87
88      delay(delay_time);
89
90          /* do the periodic stuff */
91          periodic_stuff();
92      }
93
94      /* clean things up */
95      cleanup_stuff();
96
97      return 0;
98      }
99
100
101
102 initialize_stuff(argn, argv)
103
/*******************************************************
**********
104 initialize_stuff takes care of all initialization
105
*******************************************************
*********/
106     int argn;
107     char **argv;
108     {
109     extern struct timeval start_time;
110     char *eptr;
111     key_t key;
112
113     /* system V -- ignore children who terminate */
114     signal(SIGCHLD, SIG_IGN);
115
116     /* run real-time to avoid spinlock problems */
117     if (rtprio(0, 127) < 0)
118        perror("Driver requires real-time privileges.
Continuing anyway ...");
119
120     /* get the shared memory key */
121     eptr = getenv("TPCC_SHMKEY_DRIVER");
122     if (eptr == NULL)   key = SHMKEY_DRIVER;
123     else                key = atoi(eptr);
```

```
124
125     shm = (shm_t *)attach_shm(key, sizeof(shm_t), 0);
126     if (shm == NULL) {
127        error("Could not attach to shared memory (key
 = %d).\n",key);
128     }
129
130     /* initialize the clock */
131     initclock();
132     shm->start_time = start_time;
133
134     /* parse the arguments */
135     configure(argn, argv);
136
137     /* initialize the log */
138     initialize_success(key);
139
140     /* initialize informiation file */
141     initialize_info(key);
142
143 #ifdef RAMP_UP
144     if (getenv("DRIVER_SPAWN_RATE") &&
 atoi(getenv("DRIVER_SPAWN_RATE")))
145     spawn_rate = atoi(getenv("DRIVER_SPAWN_RATE"));
146     if (getenv("DRIVER_SPAWN_PERIOD") &&
 atoi(getenv("DRIVER_SPAWN_PERIOD")))
147     spawn_period =
 atoi(getenv("DRIVER_SPAWN_PERIOD"));
148 #endif
149     }
150
151
152 periodic_stuff()
153
 /*******************************************************
 ***************
154 periodic_stuff runs periodically,  controlling the
 general flow of TPC-C
155
 *******************************************************
 **************/
156     {
157
158     /* see if we are ready to stop */
159     if ((transactions > max_transactions) ||
160     (getclock() > max_duration)       ||
161     (state == STOP))
162         {
163         stopflag = YES;
164         desired = 0;
165         }
166
167     /* write results */
168     flush_success();
169
170     /* if appropriate, spawn more users */
171 #ifdef RAMP_UP
172     if( spawned < desired)
173     {
174     static int spawn_wait=1; /* set it to 1 so we do
 it the first time */
175
176     /* we assume we are called every second... */
177
178     if (--spawn_wait==0)
179     {
180         int num_to_spawn = desired - spawned;
181         spawn_wait = spawn_period;
182
183         if (num_to_spawn > spawn_rate)
184             num_to_spawn = spawn_rate;
185
186         while (num_to_spawn--)
187             spawn_user();
188     }
189     }
190 #else
191     while (spawned < desired)
192     spawn_user();
193 #endif
194 }
195
196
197 cleanup_stuff()
198
 /*******************************************************
 ***************
199 cleanup_stuff takes care of all cleanup when a TPC run
 is finished
200
 *******************************************************
 ***************/
201     {
202     cleanup_success();
203     close(info_fd);  /* close the information file */
204     detach_shm((void *)shm);
205     }
206
207
208
209 spawn_user()
210
 /*******************************************************
 ***************
211 spawn_user creates a new TPC-C user
212
 *******************************************************
 **************/
213     {
214     int pid;
215     int index;
216     int cnt;
217     int ratio;
218
219     extern int errno;
220
221     ratio = (nclients+ndrivers-1)/ndrivers;
222
223     /* wait for most of the spawned users to get
 attached */
224     while (spawned > (10*ratio) + attached)
225        delay(.05);
226
227     /* allocate a userid for the new process */
228     lock(shm_lock);  index = spawned++;
 unlock(shm_lock);
229
230     /* create a child process */
231     pid = (dbg)?0: fork();
232     if (pid < 0)
233         syserror("Can't spawn any more children.  %d
 so far (errno=%d).\n", spawned,errno);
234
235     /* if we are the child ... */
236     if (pid == 0)
237         {
238
239         /* get a unique user id for this user */
240     userid = first_user + index;
241
242         /* randomize the random number generator */
243         RandomizeUser(userid);
244
245         /* start simulating a user */
246         user(userid);
247
248         /* exit when user is finished */
249         exit(0);
250         }
251
252     }
253
254
255 RandomizeUser(id)
```

```
256
 /*******************************************************
 ****************
257 A special version of Randomize for the TPCC driver
258    (Ensure each user has a different seed and each run
 is independent)
259
 *******************************************************
 ***************/
260     ID id;
261     {
262     char buf[1024];
263     extern struct timeval start_time;
264     long int seed = (start_time.tv_sec +
 start_time.tv_usec + id);
265     srand48(seed);
266
267     /* Put seed information in the info file */
268     lock(shm_lock);
269     sprintf(buf, "seed for user %d = %d\n", id, seed);
270     write_buf(info_fd, (void *)buf, strlen(buf));
271     unlock(shm_lock);
272     }
273
274 /* local data for each user */
275
276 generic_trans generic_transaction;
277 generic_trans *trans=&generic_transaction;
278
279 ID warehouse;
280 ID district;
281
282 int GetClientNum(uid, TD, drvr, TC, TU)
283     int uid,    /* This user                */
284     TD,     /* Total # of drivers           */
285     drvr,   /* This driver              */
286     TC,     /* Total # of clients           */
287     TU;     /* Desired # of users per driver    */
288 {
289     int clnt;
290
291     /* is it clients per driver */
292     if (TC >= TD)
293     {
294     int clients_per_driver = TC/TD;
295
296     if (TD * clients_per_driver != TC)
297         error("# of clients has to be an integral
 mulitple of # of drivers.\n");
298
299     /* Uid ranges from 0 -> maxusers */
300     /* put all clients from the same warehouse on one
 client machine */
301     clnt = 1 + (drvr - 1)*clients_per_driver +
 ((uid/10) % clients_per_driver);
302     }
303     else /* or is it drivers per client? */
304     {
305         int drivers_per_client = TD/TC;
306
307     if (TC * drivers_per_client != TD)
308         error("# of drivers has to be an integral
 mulitple of # of clients.\n");
309     clnt = 1 + (drvr - 1)/drivers_per_client;
310     }
311     return clnt;
312 }
313
314
315 user(userid)
316
 /*******************************************************
 ******
317 user controls the TPC-C session for a single user
318
 *******************************************************
 ******/
319     int userid;
320     {
321     char buf[1024];
322     int GetClientNum();
323     char *str;
324
325     if (server) /* Each driver is in effect a client
 */
326     {
327         clientnum = drivernum;
328         nclients = ndrivers;
329     }
330     else {
331         clientnum = GetClientNum(userid, ndrivers,
 drivernum, nclients, desired);
332     }
333
334     warehouse = (userid/10) + 1;
335     district  = (userid%10) + 1;
336
337     if (warehouse < 1 || warehouse > scale)
338         error("'error in calculating warehouse #;
 can't use %d.\n", warehouse);
339
340     /* Print out the mapping to standard out for
 debugging purposes
341         and to show the auditor */
342
343     lock(shm_lock);
344     sprintf(buf,"userid = %d, client = %d, warehouse =
 %d, district = %d\n",          userid, clientnum, ware-
 house, district);
345     write_buf(info_fd, (void *)buf, strlen(buf));
346     unlock(shm_lock);
347
348     /* attach to the delivery que */
349     delivery_init(userid);
350
351     /* attach to the transaction processor */
352     /* WARNING!!!  This transaction begin works with
 batch tpcc because
353         the "userid" is first.  The function that this
 calls only
354         really takes 1 argument (the userid).  We
 should fix this
355         at some point. */
356     transaction_begin(userid, warehouse, district,
 clientnum);
357     lock(shm_lock); attached++; unlock(shm_lock);
358
359
360     /* repeat until we are not needed */
361     while (userid <  first_user + desired)
362         {
363
364         /* if we are a server, select a random ware-
 house and district */
365         if (server)
366             {
367             warehouse = RandomNumber((drivernum - 1) *
 (int) (scale / ndrivers) + 1,
368              drivernum * (int) (scale / ndrivers));
369              district  = RandomNumber(1, no_dist_pw);
370         if (warehouse < 1 || warehouse > scale)
371         error("'error in calculating warehouse #;
 can't use %d.\n", warehouse);
372             }
373
374         /* if users are ramping up or down, then sleep
 */
375 #ifdef RAMP_UP
376         if (state == WAIT )
377 #else
378         if (attached != desired || state == WAIT )
379 #endif
380             sleep(1);
381
```

```
382          /* otherwise do the transaction */
383          else
384              do_C_transaction();
385          } /* end of while */
386
387      /* detach from the transaction processor */
388      transaction_done();
389
390      /* detach from the delivery queue */
391      delivery_done();
392
393      /* decrement the number of spawned users */
394      lock(shm_lock); spawned--; attached--;
 unlock(shm_lock);
395      }
396
397
398
399  int server_default;/* link in a non-zero value to
 simulate transaction server*/
400
401  configure(argc, argv)
402
 /*******************************************************
 ************
403  configure_test processes the command string and ini-
 tializes the overall test
404
 *******************************************************
 ***************/
405      int argc;
406      char **argv;
407      {
408      double atof();
409      extern char *optarg;
410      extern int optind, opterr;
411      char ch;
412      char *str;
413      int i;
414
415      ndrivers = 1;
416      drivernum = 1;
417
418      /* get the environment variables */
419      get_tpcc_env();
420
421      server = server_default;
422
423      /* while there are options */
424     while ((ch = getopt (argc, argv, "sSn:N:Z:lL")) !=
 -1)
425
426          /* process according to options */
427          switch ( ch )
428              {
429          /* check for server mode  */
430              case 'S':  server = YES;
431                         break;
432              case 's':  server = NO;
433                         break;
434              case 'n':  ndrivers = atoi(optarg);
435                         break;
436              case 'N':  drivernum = atoi(optarg);
437                         break;
438
439              case 'Z':  think_factor = atof(optarg);
440                         break;
441
442              case 'l':  post_transactions = NO;
443                 break;
444
445              case 'L':  post_transactions = YES;
446                 break;
447
448              default:   error("Bad runstring argu-
 ment.\n");
449                         break;
450              }
451
452      /* set the number of clients */
453      str = getenv("NR_CLIENT");
454      if (str != NULL)
455          nclients = atoi(str);
456      else
457          nclients = 1;
458
459      /* error if any options left over */
460      if (optind != argc)
461          error("Bad runstring argument.\n");
462      }
463
464  do_C_transaction()
465      {
466      int type;
467
468      /* get the menu choice */
469      type = get_trans_type();
470
471      /* process according to the choice */
472      switch(type)
473          {
474          case NEWORDER: neworder(&trans->neworder);
 break;
475          case PAYMENT:  payment(&trans->payment);
 break;
476          case ORDSTAT:  ordstat(&trans->ordstat);
 break;
477          case DELIVERY: delivery(&trans->delivery);
 break;
478          case STOCKLEV: stocklev(&trans->stocklev);
 break;
479          }
480      }
481
482  neworder(t)
483      neworder_trans *t;
484      {
485      TIME t1, t2, t3, t4, t5;
486
487      /* generate the transaction */
488      neworder_gen(t);
489
490      /* select the transaction and wait for form */
491      t1 = getclock();
492      neworder_select();
493
494      /* key in the transaction */
495      t2 = getclock() + m_fudge[NEWORDER];
496      neworder_key(t);
497      if (!server) delay(key[NEWORDER] +
 m_fudge[NEWORDER]);
498
499      /* get the results */
500      t3 = getclock();
501      neworder_transaction(t);
502      t4 = getclock() + t_fudge[NEWORDER];
503
504      /* accumulate the statistics */
505      quickstat(NEWORDER, t3, t4-t3, t);
506
507      /* think for a bit */
508      if (!server) RandomDelay(think[NEWORDER],
 t_fudge[NEWORDER]);
509      t5 = getclock();
510
511      /* post the results */
512      post_success(NEWORDER, t1, t2, t3, t4, t5, t);
513      }
514
515  payment(t)
516      payment_trans *t;
517      {
518
519      TIME t1, t2, t3, t4, t5;
```

```
520
521     /* generate the transaction */
522     payment_gen(t);
523
524     /* select the transaction type and wait for menu
 to be displayed */
525     t1 = getclock();  /* or start = endthink?? */
526     payment_select();
527
528     /* key in the transaction  */
529     t2 = getclock() + m_fudge[PAYMENT];
530     payment_key(t);
531     if (!server) delay(key[PAYMENT] + m_fudge[PAY-
 MENT]);
532
533     /* get the results */
534     t3 = getclock();
535     payment_transaction(t);
536     t4 = getclock() + t_fudge[PAYMENT];
537
538     /* accumulate the statistics */
539     quickstat(PAYMENT, t3, t4-t3, t);
540
541     /* think for a bit */
542     if (!server) RandomDelay(think[PAYMENT],
 t_fudge[PAYMENT]);
543     t5 = getclock();
544
545     /* post the results */
546     post_success(PAYMENT, t1, t2, t3, t4, t5, t);
547     }
548
549  ordstat(t)
550     ordstat_trans *t;
551     {
552     TIME t1, t2, t3, t4, t5;
553
554
555     /* generate the transaction */
556     ordstat_gen(t);
557
558     /* select the transaction and wait for screen */
559     t1 = getclock();
560     ordstat_select();
561
562     /* key in the transaction */
563     t2 = getclock() + m_fudge[ORDSTAT];
564     ordstat_key(t);
565     if (!server) delay(key[ORDSTAT] + m_fudge[ORD-
 STAT]);
566
567     /* get the results */
568     t3 = getclock();
569     ordstat_transaction(t);
570     t4 = getclock() + t_fudge[ORDSTAT];
571
572     /* accumulate the statistics */
573     quickstat(ORDSTAT, t3, t4-t3, t);
574
575     /* think for a bit */
576     if (!server) RandomDelay(think[ORDSTAT],
 t_fudge[ORDSTAT]);
577     t5 = getclock();
578
579     /* post the results */
580     post_success(ORDSTAT, t1, t2, t3, t4, t5, t);
581     }
582
583  delivery(t)
584     delivery_trans *t;
585     {
586     TIME t1, t2, t3, t4, t5;
587
588
589     /* generate the transaction  and send the key
 strokes */
590     delivery_gen(t);
591
592     /* select the transaction and wait for menu */
593     t1 = getclock();  /* or start = endthink?? */
594     delivery_select();
595
596     /* key in the transaction */
597     t2 = getclock() + m_fudge[DELIVERY];
598     delivery_key(t);
599     if (!server) delay(key[DELIVERY] + m_fudge[DELIV-
 ERY]);
600
601     /* get the results */
602     t3 = getclock();
603     delivery_enque(t);
604     t4 = getclock() + t_fudge[DELIVERY];
605
606     /* accumulate the statistics */
607     quickstat(DELIVERY, t3, t4-t3, t);
608
609     /* think for a bit */
610     if (!server) RandomDelay(think[DELIVERY],
 t_fudge[DELIVERY]);
611     t5 = getclock();
612
613     /* post the results */
614     post_success(DELIVERY, t1, t2, t3, t4, t5, t);
615     }
616
617  stocklev(t)
618     stocklev_trans *t;
619     {
620     TIME t1, t2, t3, t4, t5;
621
622
623     /* generate the transaction */
624     stocklev_gen(t);
625
626     /* select transaction type and wait for menu */
627     t1 = getclock();
628     stocklev_select();
629
630     /* key in the data */
631     t2 = getclock() + m_fudge[STOCKLEV];;
632     stocklev_key(t);
633     if (!server) delay(key[STOCKLEV] + m_fudge[STOCK-
 LEV]);
634
635     /* get the results */
636     t3 = getclock();
637     stocklev_transaction(t);
638     t4 = getclock() + t_fudge[STOCKLEV];
639
640     /* accumulate the statistics */
641     quickstat(STOCKLEV, t3, t4-t3, t);
642
643     /* think for a bit */
644     if (!server) RandomDelay(think[STOCKLEV],
 t_fudge[STOCKLEV]);
645     t5 = getclock();
646
647     /* post the results */
648     post_success(STOCKLEV, t1, t2, t3, t4, t5, t);
649     }
650
651  get_tpcc_env()
652
 /*******************************************************
 ****************
653  Get the environment variables
654
 *******************************************************
 ******************/
655     {
656     char *eptr;
657     long i;
658
659     eptr  = getenv("TPCCWAR");
```

```
660     if (eptr == NULL) no_warehouse = -1;
661     else              no_warehouse = atoi(eptr);
662
663     eptr  = getenv("TPCCDIS");
664     if (eptr == NULL) no_dist_pw = DIST_PER_WARE;
665     else              no_dist_pw = atoi(eptr);
666
667     eptr  = getenv("TPCCCUS");
668     if (eptr == NULL) no_cust_pd = CUST_PER_DIST;
669     else              no_cust_pd = atoi(eptr);
670
671     eptr  = getenv("TPCCORD");
672     if (eptr == NULL) no_ord_pd = ORD_PER_DIST;
673     else              no_ord_pd = atoi(eptr);
674
675     eptr  = getenv("TPCCITM");
676     if (eptr == NULL) no_item = MAXITEMS;
677     else              no_item = atoi(eptr);
678
679     eptr  = getenv("TPCCSEED");
680     if (eptr == NULL) tpcc_load_seed = LOADSEED;
681     else              tpcc_load_seed = atoi(eptr);
682
683     /*
684      * Run time values
685      */
686     eptr  = getenv("TRANS_TIME");
687     if (eptr != NULL) max_duration = atof(eptr)*60;
688
689     eptr  = getenv("TRANS_NUM");
690     if (eptr != NULL) max_transactions = atoi(eptr);
691
692     eptr  = getenv("TRANS_TYPE");
693     if (eptr != NULL) trans_type = atoi(eptr);
694
695     eptr  = getenv("OUTPUT_LEVEL");
696     if (eptr != NULL) {
697     i = atoi(eptr);
698     if (i == 1) { post_transactions = YES; }
699     }
700
701     /*
702      * Constants
703      */
704     eptr  = getenv("CLAST_CONST_C");
705     if (eptr != NULL) CLAST_CONST_C  = atoi(eptr);
706
707     eptr  = getenv("CID_CONST_C");
708     if (eptr != NULL) CID_CONST_C  = atoi(eptr);
709
710     eptr  = getenv("IID_CONST_C");
711     if (eptr != NULL) IID_CONST_C  = atoi(eptr);
712
713     /*
714      * Keying times
715      */
716     eptr  = getenv("NEWO_KEY");
717     if (eptr != NULL) key[NEWORDER] = atof(eptr);
718     eptr  = getenv("PMT_KEY");
719     if (eptr != NULL) key[PAYMENT] = atof(eptr);
720     eptr  = getenv("OS_KEY");
721     if (eptr != NULL) key[ORDSTAT] = atof(eptr);
722     eptr  = getenv("DVRY_KEY");
723     if (eptr != NULL) key[DELIVERY] = atof(eptr);
724     eptr  = getenv("STKL_KEY");
725     if (eptr != NULL) key[STOCKLEV] = atof(eptr);
726
727     /*
728      * Think time delays
729      */
730     eptr  = getenv("NEWO_THINK");
731     if (eptr != NULL) think[NEWORDER] = atof(eptr);
732     eptr  = getenv("PMT_THINK");
733     if (eptr != NULL) think[PAYMENT] = atof(eptr);
734     eptr  = getenv("OS_THINK");
735     if (eptr != NULL) think[ORDSTAT] = atof(eptr);
736     eptr  = getenv("DVRY_THINK");
737     if (eptr != NULL) think[DELIVERY] = atof(eptr);
738     eptr  = getenv("STKL_THINK");
739     if (eptr != NULL) think[STOCKLEV] = atof(eptr);
740
741     /*
742      * Menu emulated delays for Emulex if any at all
743      */
744     eptr  = getenv("NEWO_MENU");
745     if (eptr != NULL) m_fudge[NEWORDER] = atof(eptr);
746     eptr  = getenv("PMT_MENU");
747     if (eptr != NULL) m_fudge[PAYMENT] = atof(eptr);
748     eptr  = getenv("OS_MENU");
749     if (eptr != NULL) m_fudge[ORDSTAT] = atof(eptr);
750     eptr  = getenv("DVRY_MENU");
751     if (eptr != NULL) m_fudge[DELIVERY] = atof(eptr);
752     eptr  = getenv("STKL_MENU");
753     if (eptr != NULL) m_fudge[STOCKLEV] = atof(eptr);
754
755
756     /*
757      * Transmission emulated delays for Emulex if any
 at all
758      */
759     eptr  = getenv("COMM_ADJUST_NEWO");
760     if (eptr != NULL) t_fudge[NEWORDER] = atof(eptr);
761     eptr  = getenv("COMM_ADJUST_PMT");
762     if (eptr != NULL) t_fudge[PAYMENT] = atof(eptr);
763     eptr  = getenv("COMM_ADJUST_ORDS");
764     if (eptr != NULL) t_fudge[ORDSTAT] = atof(eptr);
765     eptr  = getenv("COMM_ADJUST_DVRY");
766     if (eptr != NULL) t_fudge[DELIVERY] = atof(eptr);
767     eptr  = getenv("COMM_ADJUST_STKL");
768     if (eptr != NULL) t_fudge[STOCKLEV] = atof(eptr);
769
770     }
771
772
773  /*******************************************************
 *********
774  *******************************************************
 **********
775
776  Logging Routines
777
778  *******************************************************
 *********
779  *******************************************************
 *********/
780  int s_fd = -1;
781
782
783  post_success(type, t1, t2, t3, t4, t5, t)
784
 /*******************************************************
 *****************
785  post_success adds one transaction's worth of data to
 the shmem log buffer
786
 *******************************************************
 *****************/
787     int type;
788     TIME t1, t2, t3, t4, t5;
789     generic_trans *t;
790     {
791     success_t s[1];
792     unsigned short len;
793     int err;
794
795     /* cast the pointers for each transaction */
796     neworder_trans *no = &t->neworder;
797     ordstat_trans *o = &t->ordstat;
798     payment_trans *p = &t->payment;
799
```

```
800      /* decide if we had an error or not */
801      err = (t->status != OK &&
802            !(type==NEWORDER && t->status ==
 E_INVALID_ITEM));
803
804      /* record the transaction in the log area */
805      s->type = type;
806      s->t1 = t1;
807      s->t2 = t2;
808      s->t3 = t3;
809      s->t4 = t4;
810      s->t5 = t5;
811      s->status = t->status;
812
813     /* process according to the type of transaction */
814      if (type == NEWORDER)
815          {
816          int i;
817          s->ol_cnt = no->O_OL_CNT;
818          s->remote_ol_cnt = 0;
819          for (i=0; i<no->O_OL_CNT; i++)
820              if (no->item[i].OL_SUPPLY_W_ID != no-
 >W_ID)
821                  s->remote_ol_cnt++;
822          }
823
824      else if (type == PAYMENT)
825          {
826          s->remote = (p->W_ID != p->C_W_ID);
827          s->byname = p->byname;
828          }
829
830      else if (type == ORDSTAT)
831          {
832          s->byname = o->byname;
833          }
834
835      /* figure out how much data we're going to write
 */
836      len = sizeof(*s);
837      if (err || post_transactions)     len +=
 transaction_size[type];
838
839      /* write the success record and optionally append
 the trans itself */
840      lock(shm_lock);
841      write_success(&len, sizeof(len));
842      write_success(s, sizeof(*s));
843      if (err || post_transactions)
844      write_success(t, transaction_size[type]);
845      transactions++;
846      unlock(shm_lock);
847      }
848
849
850
851  quickstat(type, time, response, t)
852
 /*******************************************************
 **********
853  quickstat accumulates quick realtime statistics
854
 *******************************************************
 **********/
855      int type;
856      TIME time, response;
857      generic_trans *t;
858      {
859
860
861      /* Log a message if we had an error. (Here is
 sooner than post_succes)*/
862      if (t->status != OK &&
863            !(type==NEWORDER && t->status ==
 E_INVALID_ITEM))
864          message("%s Error %d\n",
 transaction_name[type], t->status);
865
866      /* update the statistics (don't worry about shm
 locks?) */
867      stat->count[type]++;
868      stat->total_response[type] += response;
869      if (response > valid_response[type])
870          stat->bad[type]++;
871      }
872
873
874  void
875  initialize_info(shmkey)
876
 /*******************************************************
 **************
877  initialize_info opens the info file
878
 *******************************************************
 ****************/
879  key_t shmkey;
880  {
881      char filename[80];
882      char *s_name;
883
884      /* get the name of the s_ file */
885      s_name = getenv("TPCC_INFO_FILE");
886      if (s_name == NULL)  s_name =
 "/tmp/TPCC_INFO_FILE";
887
888      /* append the shmkey to the filename */
889      sprintf(filename,"%s.%d",s_name,shmkey);
890      s_name=filename;
891
892      /* open the s_ file */
893      info_fd = open(s_name, O_RDWR|O_CREAT|O_TRUNC,
 0666);
894      if (info_fd < 0)
895          syserror("Can't open informiation file
 '%s'\n", s_name);
896  }
897
898  initialize_success(shmkey)
899
 /*******************************************************
 **************
900  initialize_success opens the results file and resets
 the shm buffer
901
 *******************************************************
 ****************/
902      key_t shmkey;
903      {
904      char filename[80];
905      char *s_name;
906      success_header_t header;
907      extern struct timeval start_time;
908
909      /* get the name of the s_ file */
910      s_name = getenv("TPCC_SUCCESS_FILE");
911      if (s_name == NULL)  s_name =
 "/tmp/TPCC_SUCCESS_FILE";
912
913      /* append the shmkey to the filename */
914      sprintf(filename,"%s.%d",s_name,shmkey);
915      s_name=filename;
916
917      /* open the s_ file */
918      s_fd = open(s_name, O_RDWR|O_CREAT|O_TRUNC,
 0666);
919      if (s_fd < 0)
920          syserror("Can't open s_ file %s\n", s_name);
921
922      /* clear the sucess buffer */
923      S_NEXT = S_OLD = S_FIRST;
924
925      /* write a header to the file */
926      if ( drivernum == 1 )
```

```
927     {
928         header.start_time = start_time;
929         write_success(&header, sizeof(header));
930     }
931     }
932
933
934 write_success(buf, len)
935
 /*******************************************************
 *********************
936 write_success writes a buffer to the success file
 (must ensure mutual exclusion)
937
 *******************************************************
 ********************/
938     char *buf;
939     int len;
940     {
941     char *old = S_OLD;
942
943     /* copy each byte, checking for wraparound and
 overflow */
944     while (len-- > 0)
945         {
946     *S_NEXT++ = *buf++;
947     if (S_NEXT >= S_LAST)
948         S_NEXT = S_FIRST;
949         if (S_NEXT == old)
950         error("Success buffer overflowed\n");
951         }
952     }
953
954
955
956 flush_success()
957
 /*******************************************************
 *********************
958 flush_success periodically copies data from shm
 buffer to results file
959
 *******************************************************
 *********************/
960     {
961     char *next;
962
963     /* rememeber where we are.  Others can add data
 while we are writing */
964     next = S_NEXT;
965
966     if (next != S_OLD)
967         debug("flush_success: next=%d old=%d\n", next-
 S_FIRST,S_OLD-S_FIRST);
968
969     /* Case: No wraparound has occurred */
970     if      (S_OLD < next)
971         write_buf(s_fd, S_OLD, next-S_OLD);
972
973     /* Case: Wraparound has happened */
974     else if (S_OLD > next)
975         {
976         write_buf(s_fd, S_OLD, S_LAST-S_OLD);
977         write_buf(s_fd, S_FIRST, next-S_FIRST);
978         }
979
980     /* Remove the old data from the buffer */
981     S_OLD = next;
982     }
983
984
985
986 cleanup_success()
987
 /*******************************************************
 ************
988 cleanup_success flushes the final transactions and
  closes the results file
989
 *******************************************************
 ************/
990     {
991     flush_success();
992     close(s_fd);
993     }
994
995
996
997
998 static write_buf(fd, buf, len)
999
 /*******************************************************
 ***************
1000 write_buf outputs a fixed size buffer
1001
 *******************************************************
 ***************/
1002     int fd;
1003     void *buf;
1004     int len;
1005     {
1006     int actual;
1007
1008     debug("write_buf:  fd=%d  buf=0x%x  len=%d\n", fd,
 buf, len);
1009
1010     actual = write(fd, buf, len);
1011     if (actual != len)
1012         syserror("Error: Only wrote %d of %d
 bytes\n", actual,len);
1013     }
1014
```

driver/generate.c

```
1
 /*******************************************************
 *********************
2  @(#) Version: A.10.10 $Date: 97/02/27 23:27:05 $
3
4  (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
5
 *******************************************************
 *********************/
6
7 #include <stdio.h>
8 #include <values.h>
9 #include <unistd.h>
10 #include <time.h>
11 #include <sys/types.h>
12 #include <sys/ipc.h>
13 #include <fcntl.h>
14 #include <signal.h>
15 #include <math.h>
16
17 #include "shm_lookup.h"
18 #include "random.h"
19
20 #include <time.h>
21
22 int CLAST_CONST_C = 208;
23 int CID_CONST_C =  37;
24 int IID_CONST_C = 75;
25
26 int trans_type = 0;   /* type of transaction 0 == all
 */
27
28 extern ID warehouse;
29 extern ID district;
30
31 extern int no_warehouse;
32 extern int no_item;
33 extern int no_dist_pw;
```

```
 34 extern int no_cust_pd;
 35 extern int no_ord_pd;
 36 extern int no_new_pd;
 37 extern int tpcc_load_seed;
 38
 39 neworder_gen(t)
 40     neworder_trans *t;
 41     {
 42     int i;
 43
 44     t->W_ID = warehouse;
 45
 46     t->D_ID = RandomNumber(1, no_dist_pw);
 47     t->C_ID = NURandomNumber( 1023, 1, no_cust_pd,
CID_CONST_C);
 48
 49     t->O_OL_CNT = RandomNumber(5, 15);
 50
 51     for (i=0; i<t->O_OL_CNT; i++)
 52         {
 53         t->item[i].OL_I_ID = NURandomNumber(8191, 1,
no_item, IID_CONST_C);
 54         t->item[i].OL_SUPPLY_W_ID = RandomWare-
house(warehouse, scale, 1);
 55         t->item[i].OL_QUANTITY = RandomNumber(1, 10);
 56         }
 57
 58     /* 1% of transactions roll back. Give the last
order line a bad item */
 59     if (RandomNumber(1, 100) == 1)
 60         t->item[t->O_OL_CNT - 1].OL_I_ID = -1;
 61     }
 62
 63 payment_gen(t)
 64     payment_trans *t;
 65     {
 66
 67     /* home warehouse is fixed */
 68     t->W_ID = warehouse;
 69
 70     /* Random district */
 71     t->D_ID = RandomNumber(1, no_dist_pw);
 72
 73     /* Customer is from remote warehouse and district
15% of the time */
 74     t->C_W_ID = RandomWarehouse(warehouse, scale,
15);
 75     if (t->C_W_ID == t->W_ID)
 76         t->C_D_ID = t->D_ID;
 77     else
 78         t->C_D_ID = RandomNumber(1, no_dist_pw);
 79
 80     /* by name 60% of the time */
 81     t->byname = RandomNumber(1, 100) <= 60;
 82     if (t->byname)
 83        LastName(NURandomNumber(255, 0, no_cust_pd/3 -
1, CLAST_CONST_C),
 84          t->C_LAST);
 85     else
 86         t->C_ID = NURandomNumber(1023, 1, no_cust_pd,
CID_CONST_C);
 87
 88     /* amount is random from [1.00..5,000.00] */
 89     t->H_AMOUNT = RandomNumber(100, 500000);
 90
 91     }
 92
 93 ordstat_gen(t)
 94     ordstat_trans *t;
 95     {
 96
 97     /* home warehouse is fixed */
 98     t->W_ID = warehouse;
 99
100     /* district is randomly selected from warehouse */
101     t->D_ID = RandomNumber(1, no_dist_pw);
102
103     /* by name 60% of the time */
104     t->byname = RandomNumber(1, 100) <= 60;
105     if (t->byname)
106        LastName(NURandomNumber(255, 0, no_cust_pd/3 -
 1, CLAST_CONST_C),
107          t->C_LAST);
108     else
109        t->C_ID = NURandomNumber(1023, 1, no_cust_pd,
 CID_CONST_C);
110     }
111
112 delivery_gen(t)
113     delivery_trans *t;
114     {
115     t->W_ID = warehouse;
116     t->O_CARRIER_ID = RandomNumber(1,10);
117     }
118
119 stocklev_gen(t)
120     stocklev_trans *t;
121     {
122     t->W_ID = warehouse;
123     t->D_ID = district;
124     t->threshold = RandomNumber(10, 20);
125     }
126
127 int get_trans_type()
128
 /*******************************************************
 *****************
129  * get_trans_type selects a transaction according to
 the weighted average
130  * For TPC-C rev 3.0 and less and TPC-C rev 3.2 this
 is:
131  *       new-order :  ???
132  *       payment   : 43.0%
133  *       order stat:  4.0%
134  *       delivery  :  4.0%
135  *       stock     :  4.0%
136
 *******************************************************
 ****************/
137     {
138     static double weight[] = { 0.0,  0.0, .4305,
 .0405, .0405, .0405};
139     double drand48();
140     int type;
141     double r;
142
143     /* choose a random number between 0.0 and 1.0 */
144     if (trans_type == 0) {
145         r = drand48();
146
147         /*
148          * select one of STOCKLEV, DELIVERY, ORDSTAT
 and PAYMENT
149          * based on weight
150          */
151         for (type = STOCKLEV;  type > NEWORDER;  type-
 -) {
152             r -= weight[type];
153             if (r < 0) break;
154         }
155     } else {
156     /* user wants only a certain type (say all stock-
 level) so do that
157            instead */
158     type = trans_type;
159     }
160     /* return the value of the selected card, or
 NEWORDER if none selected */
161     return type;
162     }
```

driver/init_shm.c

1

```
/*******************************************************
**********************
 2  @(#) Version: A.10.10 $Date: 97/02/27 23:27:05 $
 3
 4  (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*******************************************************
**********************/
 6 #include <unistd.h>
 7 #include <time.h>
 8 #include <values.h>
 9 #include <sys/types.h>
10 #include <sys/ipc.h>
11 #include <sys/shm.h>
12 #include <fcntl.h>
13 #include <signal.h>
14 #include <math.h>
15 #include "tpcc.h"
16 #include "shm.h"
17 #include "spinlock.h"
18 #include "master.h"
19
20 int main(argn, argv)
21
/*******************************************************
**********
22 TPC-C driver program
23
*******************************************************
*********/
24      int argn;
25      char **argv;
26      {
27      /* initialize everything */
28      initialize_stuff(argn, argv);
29      return 0;
30      }
31
32 initialize_stuff(argn, argv)
33
/*******************************************************
**********
34 initialize_stuff takes care of all initialization
35
*******************************************************
*********/
36      int argn;
37      char **argv;
38      {
39      struct timeval current;
40      char *str;
41      key_t key;
42
43      /* get the shared memory key */
44      str = getenv("TPCC_SHMKEY_DRIVER");
45      if (str == NULL)   key = SHMKEY_DRIVER;
46      else               key = atoi(str);
47
48      /* create the shared memory */
49      shm = (shm_t *)attach_shm(key,sizeof(shm_t),1);
50
51      /* parse the arguments */
52      configure(argn, argv);
53
54      detach_shm(shm);
55      }
56
57 void clear_stats()
58 {
59      int i;
60      for(i = 0; i < 6; i++) {
61      stat->total_response[i] = 0.0;
62      stat->count[i]          = 0;
63      stat->bad[i]            = 0;
64      }
65 }
66
67 int server_default;/* link in a non-zero value to
simulate transaction server*/
68
69 configure(argc, argv)
70
/*******************************************************
***********
71 configure_test processes the command string and ini-
tializes the overall test
72
*******************************************************
***************/
73      int argc;
74      char **argv;
75      {
76      double atof();
77      extern struct timeval start_time;
78      extern char *optarg;
79      extern int optind, opterr;
80      char *str;
81      char ch;
82      int i;
83
84      /* define the default configuration */
85      desired          = -1;  /* defaults to 10*scale */
86      ndrivers          = 1;
87      drivernum         = 1;
88      state             = WAIT;
89      stopflag          = NO;
90      attached          = 0;
91      spawned           = 0;
92      transactions      = 0;
93      first_user        = 0;
94      max_transactions  = 0x7fffffff;
95      max_duration      = 9000.0;  /* 2.5 hours */
96      fudge             = -1.0;
97      dbg               = NO;
98      think_factor      = 1.05;
99      post_transactions = NO;
100
101     clear_stats();
102
103     /* set up the lock */
104     init_lock(shm_lock);
105
106     /* initialize the clock */
107     initclock();
108     shm->start_time = start_time;
109
110     /*
111      * Run time values
112      */
113     str  = getenv("TRANS_TIME");
114     if (str != NULL) max_duration = atof(str)*60;
115
116     str  = getenv("TRANS_NUM");
117     if (str != NULL) max_transactions = atoi(str);
118
119     scale             = no_warehouse;
120     server            = server_default;
121
122     /* while there are options */
123     while ((ch = getopt (argc, argv,
 "S:s:n:N:f:F:i:w:T:u:I:DZ:Ll")) != -1)
124
125         /* process according to options */
126         switch ( ch )
127             {
128
129         /* check for server mode  */
130             case 'S':  server = YES;
131                        break;
132             case 's':  server = NO;
133                        break;
134             case 'n':  ndrivers = atoi(optarg);
135                        break;
```

```
136             case 'N':  drivernum = atoi(optarg);
137                        break;
138
139             /* get the fudge factor, if any (seconds)
 */
140             case 'f':  fudge = atof(optarg);
141                        break;
142
143             /* get the user id of the first user */
144          case 'F':  first_user = atoi(optarg);
145                 break;
146
147            /* find how many transactions to perform */
148             case 'i':  max_transactions =
 atoi(optarg);
149                        break;
150
151             /* get the scaling factor */
152             case 'w':  scale = atoi(optarg);
153                        break;
154
155             /* get the total run time (minutes) */
156             case 'T':  max_duration = atof(optarg) *
 60;
157                        break;
158
159             /* get the number of users */
160             case 'u':  desired = atoi(optarg);
161                        break;
162
163             case 'I':  state = atoi(optarg);
164                        break;
165
166             case 'D':  dbg = YES;
167                        desired = 1;
168                        scale = 5;
169                        break;
170
171             case 'Z':  think_factor = atof(optarg);
172                        break;
173
174             case 'l':  post_transactions = NO;
175                break;
176
177             case 'L':  post_transactions = YES;
178                break;
179
180             default:   error("Bad runstring argument
 [%c].\n", ch );
181                        break;
182             }
183
184     /* error if any options left over */
185     if (optind != argc)
186          error("Bad runstring argument, optind !=
 argc\n");
187
188     /* set up any remaining defaults */
189     if (!server)
190     {
191         if (desired == -1 && scale != -1)   desired =
 scale*10;
192         if (scale == -1   && desired != -1) scale =
 (desired+9)/10;
193         if (fudge == -1.0)                  fudge =
 0.305;
194     }
195     else
196     {
197     if (desired == -1 && scale != -1)   desired =
 (scale+4) / 5;
198     if (fudge == -1.0)                  fudge = 0.0;
199     }
200
201     i = desired/ndrivers ;
202     desired = i + desired % i;
203
204     if (drivernum == 1) {
205        /* create the master shared memory only on the
 master driver                        number. (needed so
 stats can read it) */
206          shm = (shm_t
 *)attach_shm(SHMKEY_MASTER,sizeof(shm_t),1);
207         /* No need to initialize, master will do that
 */
208         first_user = 0;
209     } else {
210     /* take into account that the master driver might
 have more
211            users on it, first_user on the other driv-
 ers is
212            #users on first driver + multiples of the
 count
213           of users on all the other drivers.  At this
 point
214            drivernum is at least 2 */
215         first_user = desired + (drivernum - 2)*i;
216         desired    = i;
217     }
218
219     /* make sure parameters are within range */
220     if (state != WAIT && state != RUN && state !=
 STOP)
221         error("'state' option is out of range.\n");
222     if (transactions < 0)
223         error("'transactions' option is out of
 range.\n");
224  }
```

driver/keystroke_web.c

```
 1
/*********************************************************
**********************
 2   @(#) Version: A.10.10 $Date: 97/02/27 23:27:05 $
 3
 4   (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*********************************************************
**********************/
 6
/*********************************************************
**********************
 7  History
 8  941013 JVM Added an audit string to the login form
 9
10
*********************************************************
*********************/
11  #include <stdio.h>
12  #include <ctype.h>
13  #include <stdlib.h>
14  #include <signal.h>
15  #include <stdarg.h>
16  #include <string.h>
17
18  #include "tpcc.h"
19
20  #define TIME_OUT 300  /* seconds */
21
22  #define PRINT_WIDTH 60
23
24  #define ERR_TYPE_WEBDLL            3
25  #define ERR_NEWORDER_ITEMID_INVALID 1030
26  #define MAX_RETRIES 4
27
28  #define USE_PERSISTENT_CONNECTIONS
29
30  #define DEBUG_NONE   0
31  #define DEBUG_LITTLE     1
32  #define DEBUG_SOME   5
33  #define DEBUG_ALL   10
34
```

```
/* #define DEBUG DEBUG_LITTLE /* */
/* #define DEBUG DEBUG_SOME /* */
/* #define DEBUG DEBUG_ALL /* */

#ifndef DEBUG
#define DEBUG DEBUG_NONE /* default is no debugging */
#endif

#define NO_ID       -1
#define LOGIN_ID     1
#define MENU_ID      2
#define NEWORDER_ID      3
#define PAYMENT_ID   4
#define DELIVERY_ID      5
#define ORDSTAT_ID   6
#define STOCKLEVEL_ID    7

#define SMALL_BUFFER_SIZE   2048
#define BIG_BUFFER_SIZE     (SMALL_BUFFER_SIZE*4)

static char html_receive_buffer[BIG_BUFFER_SIZE];
static char html_send_buffer[BIG_BUFFER_SIZE];
static char xmit_buffer[BIG_BUFFER_SIZE];
time_t send_time;
static int warehouse;
static int district;
static int user;
static int client;
static int STATUSID=0;
static int TERMID=0;
static int SYNCID=0;
static int socket_fd=-1; /* the -1 will get us EBADF below... */

void
handle_alarm(int sig)
{
#ifdef _DEBUG
     char *time_buffer = ctime(&send_time);

     time_buffer[strlen(time_buffer)-1] = '\0'; /* ditch the newline */

     message("No response from client. send_time=%s warehouse=%d "
             "district=%d html_send_buffer=[%s]\n",
                time_buffer, warehouse,    district, html_send_buffer);
#endif
}

int
find_value(const char *const html, const char *const tag,
     char *value, const int buffer_len)
{
     int iReturn = 0;
     char *pos = strstr(html, tag);

     /* the tag was found, now get its value */
     if (pos && (pos = strstr(pos, "VALUE=\"")))
     {
         char *end_quote;

         pos += 7; /* skips over VALUE=" */
         end_quote = strchr(pos, '"');

         if (end_quote)
         {
             while(pos != end_quote)
                 *value++ = *pos++;
             *value = '\0';
             iReturn = 1;
         }
     }

     return iReturn;
}

int
get_html_int(const char *const html, const char *const tag)
{
     char buf[1024];

     if (!find_value(html, tag, buf, sizeof(buf)))
         error("get_html_int failed, html=[%s], tag=[%s]\n",
                html, tag);

     return atoi(buf);
}

/*
 * send_html_request sends length bytes to socket_fd from html_send_buffer.  If
 * it is unable to do so, it eventually calls exit(), so it is
 * declared void...
 */

void
send_html_request(void)
{
     int bytes_written;
     int length = strlen(html_send_buffer);

     bytes_written = write(socket_fd, html_send_buffer, length);

     /*
      * see if we got EBADF.  If so, assume that it was because the
      * "persistent" connecton went away.  reconnect
      * and fall through.  If it was any other error it will
      * recur inside the loop and we will stop.
      * Ain't statelessness great :-(
      */

     if (bytes_written < 0)
     {
         if (errno == EBADF)
         {
             close(socket_fd);
             socket_fd = connect_to_server(user, client);
             bytes_written=0;
#if DEBUG > 4
             message("send_html_request: connected fd=%d\n", socket_fd);
#endif
         }
         else
             syserror("send_html_request: write failed");
     }

     while(bytes_written < length)
     {
         char *write_pos = html_send_buffer+bytes_written;
         int nwritten = write(socket_fd, write_pos, length - bytes_written);

         if (nwritten < 0)
             syserror("send_html_request: write failed\n");

         bytes_written += nwritten;
     }

#if DEBUG > 4
```

```
170      message("send_html_request: wrote %d\n",
 bytes_written);
171  #endif
172
173  }
174
175  int
176  get_html_reply()
177  {
178      char *pos = html_receive_buffer;
179      int length=0;
180      int nread;
181      int retries=0;
182      extern int errno;
183
184  #if DEBUG > 9
185  message("about to read, socket_fd=%d\n", socket_fd);
186  #endif
187      while((nread = read(socket_fd, pos,
 BIG_BUFFER_SIZE - length)) >= 0
188          || errno == EINTR)
189      {
190  #if DEBUG > 4
191          message("get_html_reply(), nread = %d\n",
 nread);
192  #endif
193         if (nread <= 0) /* our "persistent" connection
 went away */
194          {
195              if (length > 0)
196              {
197                  error("get_html_reply: nread == 0,
 length=%d set=[%s], received=[%s]\n",
198                          length, html_send_buffer,
 html_receive_buffer);
199              }
200              if (retries++ > MAX_RETRIES)
201              {
202                 char *time_buffer = ctime(&send_time);
203
204                  time_buffer[strlen(time_buffer)-1] =
 '\0'; /* ditch the newline */
205
206                  error("get_html_reply: unable to re-
 establish connection. send_time=%s warehouse=%d dis-
 trict=%d html_send_buffer=[%s]\n",
207                          time_buffer, warehouse,
208                          district, html_send_buffer);
209              }
210              close(socket_fd);
211              socket_fd = connect_to_server(user, cli-
 ent);
212  #if DEBUG > 4
213              message("get_html_reply(): reconnected,
 fd=%d\n", socket_fd);
214  #endif
215            send_html_request(); /* resend the request
 */
216          }
217          else
218          {
219              pos += nread;
220              length += nread;
221              if (length >= BIG_BUFFER_SIZE)
222                  error("get_html_reply() buffer over-
 run\n");
223
224              if (*(pos-1) == '\0')
225                  break; /* the HTML will be null ter-
 minated */
226          }
227      }
228
229  #if DEBUG > 4
230      message("get_html_reply: read %d total %d\n",
 nread, length);
231  #endif
232      if (nread < 0)
233          syserror("read error in get_html_reply\n");
234
235      if (length == 0)
236          error("get_html_reply(): zero length
 read\n");
237  #if DEBUG > 9
238      message("get_html_reply: read %d bytes [%s]\n",
 length, html_receive_buffer);
239  #endif
240
241  }
242
243
244  int
245  html_transaction(const char *cmd, const int id, const
 char *format, ...)
246  {
247      char *pos = html_send_buffer;
248      int bytes_written=0;
249      extern int errno;
250  #define TRANS_SEQ
251  #ifdef TRANS_SEQ
252      static int seq=0;
253      extern int userid;
254  #endif
255
256      va_list args;
257      va_start(args, format);
258
259      pos += sprintf(pos, "GET /tpcc.dll?");
260      if (id != NO_ID)
261          pos += sprintf(pos,
262                  "STATUSID=%d&TERMID=%d&SYNCID=%d&FOR-
 MID=%d&",
263                  STATUSID, TERMID, SYNCID, id);
264  #ifdef TRANS_SEQ
265      seq++;
266      seq &= 0xffff;
267      seq |= (userid << 16);
268
269      pos += sprintf(pos, "SEQ=0x%x&", seq);
270  #endif
271
272      pos += vsprintf(pos, format, args);
273
274      pos += sprintf(pos, "CMD=%s ", cmd);
275
276      pos += sprintf(pos, "HTTP/1.0\n");
277  #ifdef USE_PERSISTENT_CONNECTIONS
278      pos += sprintf(pos, "Connection: Keep-Alive\n");
279  #endif
280      pos += sprintf(pos, "\n");
281
282      if (pos - html_send_buffer > BIG_BUFFER_SIZE)
283          syserror(stderr,"html_transaction() buffer
 overrun\n");
284
285  #if DEBUG > 9
286      message("html_transacton: buffer=[%s]\n",
 html_send_buffer);
287  #endif
288      alarm(TIME_OUT);
289      time(&send_time);
290      send_html_request();
291      get_html_reply();
292      alarm(0);
293
294      STATUSID = get_html_int(html_receive_buffer,
 "\"STATUSID\"");
295      TERMID = get_html_int(html_receive_buffer,
 "\"TERMID\"");
296      SYNCID = get_html_int(html_receive_buffer,
 "\"SYNCID\"");
297  }
298
299  int
```

```
300  tabprint(int tablevel, int width, char *buf, char
 *string)
301  {
302      int remaining=strlen(string);
303      char *orig=buf;
304
305      while(remaining)
306      {
307          int i;
308          int len;
309
310          *buf++ = '\n';
311          for(i=0;i<tablevel;i++)
312              *buf++ = '\t';
313          len = width;
314          if (len > remaining)
315              len = remaining;
316          strncpy(buf, string, len);
317          buf += len;
318          string += len;
319          remaining -= len;
320      }
321
322      return buf - orig;
323  }
324
325  check_statusid(const char *format, ...)
326  {
327      if (STATUSID != 0)
328      {
329          int len;
330          char buf[BIG_BUFFER_SIZE];
331          va_list args;
332          va_start(args, format);
333          len = vsprintf(buf, format, args);
334          len += sprintf(buf+len, " STATUSID=%d TER-
 MID=%d SYNCID=%d",
335              STATUSID, TERMID, SYNCID);
336          len += sprintf(buf+len, "\n\tSend
 Buffer:\n");
337          len += tabprint(2, PRINT_WIDTH, buf+len,
 html_send_buffer);
338          len += sprintf(buf+len, "\n\tReceive
 Buffer:\n");
339          len += tabprint(2, PRINT_WIDTH, buf+len,
 html_receive_buffer);
340          error("%s\n", buf);
341      }
342  }
343
344
345  /*
346   * in login we figure out which of the web clients to
 connect to, and
347   * then log in.
348   */
349
350
351  login()
352  {
353      char *server = getenv("SERVER");
354
355      if (server == NULL)
356          error("SERVER environment variable not
 set\n");
357
358  #if DEBUG > 4
359      message("about to connect to Server %s\n",
 server);
360  #endif
361
362      html_transaction("Begin", NO_ID, "Server=%s&",
 server);
363      check_statusid("Bad status in login() for server
 %s", server);
364
365      html_transaction("Submit", LOGIN_ID,
 "w_id=%d&d_id=%d&", warehouse, district);
366      check_statusid("Submit failed in login()");
367  #if DEBUG > 0
368      message("login completed, TERMID=%d SYNCID=%d\n",
 TERMID, SYNCID);
369  #endif
370  }
371
372  transaction_begin(usr, w_id, d_id, clnt)
373      ID w_id;
374      ID d_id;
375      int usr;
376      int clnt;
377  {
378
379      /* set an alarm handler to detect hung users */
380      signal(SIGALRM, handle_alarm);
381
382      /* decide which warehouse and district we are */
383      warehouse = w_id;
384      district =  d_id;
385      user = usr;
386      client = clnt;
387
388      /* send the login information */
389      login();
390  }
391
392
393  transaction_done()
394  {
395      html_transaction("..Exit..", MENU_ID, "");
396      check_statusid("html_transaction failed in
 transaction_done()");
397
398      close(socket_fd);
399  }
400
401
402  /*
403   * New Order
404   */
405
406  neworder_select()
407  {
408      html_transaction("..NewOrder..", MENU_ID, "");
409      check_statusid("html_transaction failed in
 neworder_select()");
410  }
411
412
413  neworder_key(t)
414      neworder_trans *t;
415  {
416      int i;
417      char *pos=xmit_buffer;
418
419      if (warehouse != t->W_ID)
420          error("warehouse  changed");
421
422      pos += sprintf(pos, "PI*=&DID*=%d&CID*=%d&", t-
 >D_ID, t->C_ID);
423
424      /* do for each order line */
425      for (i=0; i<t->O_OL_CNT; i++)
426          pos += sprintf(pos,
 "SP%.2d*=%d&IID%.2d*=%d&Qty%.2d*=%d&",
427              i, t->item[i].OL_SUPPLY_W_ID,
428              i, t->item[i].OL_I_ID,
429              i, t->item[i].OL_QUANTITY);
430
431      for(;i<15;i++) /* Yech, but there isn't a con-
 stant... */
432          pos += sprintf(pos,
 "SP%.2d*=&IID%.2d*=&Qty%.2d*=&",
433              i, i, i);
434
```

```
435  }
436
437
438  neworder_transaction(t)
439       neworder_trans *t;
440  {
441  #define ERR_TYPE_WEBDLL            3
442       html_transaction("Process", NEWORDER_ID,
 xmit_buffer);
443
444       if (STATUSID != ERR_TYPE_WEBDLL)
445            check_statusid("html_transaction failed in
 neworder_transaction()");
446
447       /* get the return status and order id */
448       get_status(&t->status);
449       if (t->status == OK)
450            get_number("Order Number:", &t->O_ID);
451  }
452
453
454
455  payment_select()
456  {
457       html_transaction("..Payment..", MENU_ID, "");
458       check_statusid("html_transaction failed in
 payment_select()");
459  }
460
461
462  payment_key(t)
463       payment_trans *t;
464  {
465       char *pos = xmit_buffer;
466
467       pos += sprintf(pos, "PI*=&DID*=%d&", t->D_ID);
468
469       if (t->byname)
470            pos += sprintf(pos,"CID*=&CLT*=%s&", t-
 >C_LAST);
471       else
472            pos += sprintf(pos, "CID*=%d&CLT*=&", t-
 >C_ID);
473
474       pos += sprintf(pos, "CWI*=%d&CDI*=%d&", t-
 >C_W_ID, t->C_D_ID);
475
476       pos += sprintf(pos, "HAM*=%.2f&", t-
 >H_AMOUNT/100);
477  }
478
479  payment_transaction(t)
480       payment_trans *t;
481  {
482       html_transaction("Process", PAYMENT_ID,
 xmit_buffer);
483       check_statusid("html_transaction failed in
 payment_transaction()");
484
485       /* get the return status */
486       get_status(&t->status);
487  }
488
489
490  ordstat_select()
491  {
492       html_transaction("..Order-Status..", MENU_ID,
 "");
493       check_statusid("html_transaction failed in
 ordstat_select()");
494  }
495
496
497
498  ordstat_key(t)
499       ordstat_trans *t;
500  {
501       char *pos = xmit_buffer;
502
503       pos += sprintf(pos, "PI*=&DID*=%d&", t->D_ID);
504
505       if (t->byname)
506            pos += sprintf(pos, "CID*=&CLT*=%s&", t-
 >C_LAST);
507       else
508            pos += sprintf(pos, "CID*=%d&CLT*=&", t-
 >C_ID);
509
510  }
511
512
513  ordstat_transaction(t)
514       ordstat_trans *t;
515  {
516       html_transaction("Process", ORDSTAT_ID,
 xmit_buffer);
517       check_statusid("html_transaction failed in
 ordstat_transaction()");
518
519       /* get the return status */
520       get_status(&t->status);
521  }
522
523
524  delivery_select()
525  {
526       html_transaction("..Delivery..", MENU_ID, "");
527       check_statusid("html_transaction failed in
 delivery_select()");
528  }
529
530
531  delivery_key(t)
532       delivery_trans *t;
533  {
534       char *pos = xmit_buffer;
535
536       pos += sprintf(pos, "PI*=&OCD*=%d&", t-
 >O_CARRIER_ID);
537  }
538
539
540
541  delivery_enque(t)
542       delivery_trans *t;
543  {
544       html_transaction("Process", DELIVERY_ID,
 xmit_buffer);
545       check_statusid("html_transaction failed in
 delivery_enqueue()");
546       /* get the return status */
547       get_status(&t->status);
548  }
549
550  delivery_init(id)
551     int id;
552  {
553  }
554
555  delivery_done()
556  {
557  }
558
559
560  stocklev_select()
561  {
562      html_transaction("..Stock-Level..", MENU_ID, "");
563       check_statusid("html_transaction failed in
 stocklev_select()");
564  }
565
566  stocklev_key(t)
567       stocklev_trans *t;
568  {
```

```
569       char *pos = xmit_buffer;
570
571       pos += sprintf(pos, "PI*=&TT*=%d&", t->thresh-
 old);
572   }
573
574
575   stocklev_transaction(t)
576       stocklev_trans *t;
577   {
578       html_transaction("Process", STOCKLEVEL_ID,
 xmit_buffer);
579       check_statusid("html_transaction failed in
 stocklev_transaction()");
580       get_status(&t->status);
581   }
582
583
584   /*
585    * Routines to extract values from the screen
586    */
587
588   get_status(status)
589       int *status;
590   {
591       char *msg;
592
593       /* if Invalid item, return the appropriate error
 */
594       if (STATUSID == ERR_TYPE_WEBDLL)
595       {
596           int err;
597           get_number("Error: TPCCWEB(", &err);
598
599           if (err == ERR_NEWORDER_ITEMID_INVALID)
600           {
601   #if DEBUG > 9
602               message("get_status(), Item number is not
 valid\n");
603   #endif
604               *status = E_INVALID_ITEM;
605               return;
606           }
607       }
608
609       if (STATUSID != 0)
610       {
611           /* something unexpected went wrong.  return
 error */
612           *status = E;
613
614   #if DEBUG > 9
615           message("get_status(), STATUSID=%d\n", STA-
 TUSID);
616   #endif
617           return;
618       }
619
620       /* Nothing major went wrong, assume status is OK
 */
621       *status = OK;
622
623       /* look for the status message in the buffer */
624       msg = strstr(html_receive_buffer, "Execution Sta-
 tus:");
625
626       /* if status not displayed, then OK */
627       if (!msg)
628       {
629   #if DEBUG > 9
630           message("get_status(), execution message not
 found in [%s]\n",
631               html_receive_buffer);
632   #endif
633           return;
634       }
635
636       /* if commited or enqueued, then OK */
637       if (strstr(msg, "Transaction commited"))
638       {
639   #if DEBUG > 9
640           message("get_status(), Transaction Com-
 mited\n");
641   #endif
642           return;
643       }
644       if (strstr(msg, "Delivery has been queued"))
645       {
646   #if DEBUG > 9
647           message("get_status(), Delivery Queued\n");
648   #endif
649           return;
650       }
651
652       /* othewise, return the generic error code */
653   #if DEBUG >= 0
654       message("get_status() returning generic error for
 [%s] send=[%s], receive=[%s]\n", msg, html_send_buffer,
 html_receive_buffer);
655   #endif
656       *status = E;
657   }
658
659   get_number(header, n)
660       char *header;
661       int *n;
662   {
663       char *p = strstr(html_receive_buffer, header);
664
665       if (!p)
666           error("problems in get_number.  header=[%s],
 html_receive_buffer=[%s]\n",
667               header, html_receive_buffer);
668
669       p += strlen(header);
670       if (sscanf(p, "%d", n) != 1)
671           error("sscanf problem in get_number with
 [%s]\n", p);
672   #if DEBUG > 9
673       message("get_number: offset=%d, number=%d,
 header=[%s]\n",
674           p - html_receive_buffer, *n, header);
675   #endif
676   }
```

driver/Makefile

```
 1  debug= +O2 +Ofastaccess +Oentrysched
 2  #debug= -w -g
 3
 4  I=../lib
 5  L=../lib
 6  Q=../msgque
 7
 8  CFLAGS= ${debug}  -I${I} -I$(ADAINC) -Wl,-a,archive
+DA1.0 -DRAMP_UP
 9  LIBS= $L/tpc_lib.a -lm
10  LDFLAGS= ${debug} ${LIBS} -lm  -Wl,-a,archive
11  PROGRAMS = driver.web qualify master slave init_shm
 convert_iis_delilog
12  ANSI_OBJS=convert_iis_delilog.o keystroke_web.o
13
14  all: ${PROGRAMS}
15
16  ${ANSI_OBJS}:
17      cc -c -Ae ${CFLAGS} $<
18
19  driver.web: driver.o  keystroke_web.o socket.o
 $L/tpc_lib.a generate.o
20      cc ${CFLAGS} ${LDFLAGS} driver.o keystroke_web.o
 socket.o generate.o ${LIBS} -o driver.web
21      ! chown root $@
22      ! chmod -w,u+s $@
23
```

```
24  qualify: qualify.o $L/tpc_lib.a
25      cc ${CFLAGS} ${LDFLAGS} qualify.o ${LIBS} -o
 qualify
26
27  connect:  connect.o  keystroke.o socket.o
 $L/tpc_lib.a
28      cc ${CFLAGS} ${LDFLAGS} connect.o keystroke.o
 socket.o ${LIBS} -o connect
29
30  convert_iis_delilog: convert_iis_delilog.o
31      cc ${CFLAGS} ${LDFLAGS} -Ae convert_iis_delilog.o
 ${LIBS} -o $@
32
33  master:  master.o $L/tpc_lib.a ../lib/master.h
34      cc ${CFLAGS} ${LDFLAGS} master.o ${LIBS} -o mas-
 ter
35
36  slave:  slave.o $L/tpc_lib.a
37      cc ${CFLAGS} ${LDFLAGS} slave.o ${LIBS} -o slave
38
39  init_shm:  init_shm.o $L/tpc_lib.a
40      cc ${CFLAGS} ${LDFLAGS} init_shm.o ${LIBS} -o
 init_shm
41
42  clean:
43      rm -f *.o
44      rm -f *.a
45      rm -f ${PROGRAMS}
46
47  clobber:  clean
48      rm -f ${PROGRAMS}
49
50  ${L}/tpc_lib.a:
51      cd ../lib; make
52
53  install: ${PROGRAMS}
54      mv ${PROGRAMS} ${WORK_DIR}/bin
55
56
```

driver/master.c

```
1
/*****************************************************
**********************
 2  @(#) Version: A.10.10 $Date: 97/02/27 23:27:05 $
 3
 4  (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*****************************************************
*********************/
 6
 7  #include <sys/types.h>
 8  #include <sys/socket.h>
 9  #include <sys/ioctl.h>
10  #include <sys/ipc.h>
11  #include <netinet/in.h>
12  #include <netdb.h>
13  #include <signal.h>
14  #include <stdio.h>
15  #include <ctype.h>
16  #include <pwd.h>
17  #include "shm.h"
18  #include "master.h"
19
20  void wait_all_done();
21
22  volatile int force_done=0;
23
24  #ifdef TKJOS_DEBUG
25  #define LOG(X) { \
26      FILE *x; \
27      x=fopen("/tmp/master.log","a"); \
28      fprintf X ;\
29      fclose(x); }
30  #else
31  #define LOG(X)
32  #endif
33
34  int users_per_driver[MAX_DRIVERS];
35
36  void
37  catch_interrupt(sig)
38  int sig;
39  {
40    printf("Master process (pid = %d) caught interrupt,
stopping run...\n",
41      getpid());
42
43     if (m_state == STOP)
44      force_done = 1;
45     else
46     {
47         m_state = STOP;
48         m_desired = 0;
49     }
50  }
51
52  void
53  catch_alarm(sig)
54  int sig;
55  {
56     printf("Master process (pid = %d) timeout
expired\n",  getpid());
57     force_done=1;
58  }
59
60  int main(argc, argv)
61
62
/*****************************************************
**********
63  TPCC information sharing program.
64
65  Input: Number of drivers and Number of users required
66
*****************************************************
*********/
67
68  int argc;
69  char **argv;
70  {
71
72      struct servent *servent;
73      int num_drivers, num_users;
74      char *USERS = getenv("DRIVER_USERS");
75
76          if ( argc < 3 )
77          {
78             fprintf(stderr,"Usage: %s <num-of-drivers>
<num_of_users> [<hostname> ... ]\n",argv[0]);
79              exit(1);
80          }
81          else
82          {
83              num_drivers = atoi(argv[1]);
84              num_users = atoi(argv[2]);
85          }
86
87      if (num_drivers > MAX_DRIVERS)
88      {
89          fprintf(stderr, "Error: num_drivers=%d,
MAX_DRIVERS=%d\n", num_drivers, MAX_DRIVERS);
90          exit(1);
91      }
92
93      memset(users_per_driver, 0,
sizeof(users_per_driver));
94      /* check if users-per-driver is specified */
95      if (USERS)
96      {
97          char *tok;
98          int n_users = 0;
```

```
 99
100         tok = strtok(USERS, "   ");
101         while (tok)
102         {
103             users_per_driver[n_users++] = atoi(tok);
104             tok = strtok(NULL, "    ");
105         }
106
107         if (n_users != num_drivers )
108         {
109            fprintf(stderr, "DRIVER_USERS specified %d
 drivers, but there are %d\n",
110                  n_users, num_drivers);
111             exit(1);
112         }
113
114     }
115
116         /* initialize everything */
117
118         initialize_stuff(argc, argv);
119
120         servent = getservbyname("exec", "tcp");
121
122 LOG ( (x,"exec_cmd(%d)...\n",num_drivers) );
123         exec_cmd(num_drivers,servent->s_port, argc,
 argv);
124
125     /* install ctrl-C catching routine */
126 LOG ( (x,"signal...\n") );
127     signal(SIGINT, catch_interrupt);
128
129 LOG ( (x,"attach_users(%d)...\n",num_drivers) );
130         attach_users(num_drivers);
131
132 LOG ( (x,"wait_all_done(%d)...\n",num_drivers) );
133     wait_all_done(num_drivers);
134
135 LOG ( (x,"cleanup_shm(%d)...\n",num_drivers) );
136         cleanup_shm(num_drivers);
137
138         exit(0);
139 }
140
141
 /*******************************************************
 ****
142
143 Executes the "slave" on the driver machines.
144 The server and slave will be xchanging data
145 over the socket connection.
146
147
 *******************************************************
 ***/
148
149 exec_cmd(n_drivers,port, argc, argv)
150 int n_drivers;
151 int port;
152 int argc;
153 char **argv;
154 {
155
156     int i;
157     char *host, buf[64];
158     char cmd[1024];
159     int  arg_index;
160
161     strcpy(cmd, argv[0]);
162     if (strrchr(cmd,'/') == NULL)
163        sprintf(cmd,"%s/bin/%s", getenv("HOME"), CMD);
164     else
165         strcpy(strrchr(cmd,'/')+1, CMD);
166
167     arg_index=strlen(cmd);
168
169     for ( i=0; i < n_drivers; i++ )
170     {
171
172           if (argc == (3 + n_drivers))
173               host = argv[3 + i ];
174           else
175           {
176               host = getenv("DRIVER");
177               if (host == NULL) host = "driver";
178                   sprintf(buf, "%s%d", host, i+1);
179               host = buf;
180           }
181
182 #define MY_SHMKEY 55556
183           sprintf(&cmd[arg_index]," %d",(MY_SHMKEY+i)
 );
184
185
186 {
187 FILE *x;
188 x=fopen("/tmp/master.log","a");
189 fprintf(x,"Master: rexec %s:%d %s\n",host,port,cmd);
190 fprintf(x,"Master: SHMKEY_DRIVER=%d  MY_SHMKEY=%d
 key=%d\n",SHMKEY_DRIVER,MY_SHMKEY,MY_SHMKEY+i);
191 fclose(x);
192 }
193           sd[i] = rexec(&host, port, NULL, NULL, cmd,
 0);
194
195 LOG( (x,"rexec done.  sd[%d]=%d\n",i,sd[i]) );
196           if ( sd[i] < 0 )
197                Error("rexec Error");
198     }
199 }
200
201
202 void
203 wait_all_done(n_drivers)
204 int n_drivers;
205 {
206     int i,ret;
207
208     signal(SIGALRM, catch_alarm);
209    alarm(5*60); /* set a timeout in case some drivers
 have died */
210 #ifndef XXX
211     fprintf(stderr, "alarm set\n");
212 #endif
213
214     while (m_attached != 0 && !force_done) {
215         for (i=0; i < n_drivers; i++) {
216
217             d_state    = STOP;
218             d_desired  = 0;
219
220 LOG( (x,"write(%d): STOP\n",i) );
221             ret =
 write(sd[i],&xfer_data,sizeof(xfer_data));
222             if (ret < 0 ) {
223           fprintf(stderr,"Ouch!!! couldn't write to
 driver %d\n",i);
224                   Error("Shared memory write");
225             }
226
227             ret =
 read(sd[i],&xfer_data,sizeof(xfer_data));
228             if (ret < 0 ) {
229           fprintf(stderr,"Ouch!!! couldn't read from
 driver %d\n",i);
230                 Error("Shared memory read");
231             }
232 LOG( (x,"read(%d): m_attached=%d\n",i,m_attached) );
233             save_stat(i);
234         }
235        delay(0.10);
236        update_shm();
237     }
238     alarm(0); /* unset timeout */
```

```
#ifndef XXX
    fprintf(stderr, "alarm canceled\n");
#endif
}


/*******************************************************
****
The main routine which decides the state for the drivers and
syncronizes the action.

*******************************************************
***/

attach_users(n_drivers)
int n_drivers;
{
    int i,reqd,ret;

    m_transactions = 0;  /* make sure these are zero to start */
    m_attached     = 0;
    while ( m_state != STOP ) {
        for ( i=0; i < n_drivers; i++ ) {

            if (users_per_driver[i] != 0) {
                reqd = users_per_driver[i];
            } else {
                reqd = m_desired/n_drivers ;
        /* allocate any partial to the first one */
            if ( i == 0 )
            reqd +=  m_desired - reqd*n_drivers;
        }

           if (m_transactions > m_max_transactions ||
        getclock() > m_max_duration ||
        m_desired == 0) {
                m_state = STOP;
                m_desired = 0;
            }
            if (m_attached == m_desired ) {
                d_state = m_state;
            } else if (m_attached != m_desired) {
#ifdef RAMP_UP
                d_state = RUN;
#else
                d_state = WAIT;
#endif
            }

            d_desired    = reqd;
            d_start_time = start_time;

            ret =
write(sd[i],&xfer_data,sizeof(xfer_data));
            if (ret < 0 ) {
        fprintf(stderr,"Ouch!!! couldn't write to driver %d (1)\n",i);
        Error("Shared memory write"); }
        LOG( (x,"write(%d): m_state=%d d_desired=%d\n",i,m_state,d_desired) );

            ret =
read(sd[i],&xfer_data,sizeof(xfer_data));
        LOG( (x,"read(%d): d_attached=%d d_state=%d\n",i,d_attached,d_state) );
            if (ret < 0 ) {
        fprintf(stderr,"Ouch!!! couldn't read from driver %d (1)\n",i);
        Error("Shared memory read"); }

            save_stat(i);
        }
    delay(0.10);   /* update 10 times per second */

        update_shm();

    }
}

Error(errstring)
char *errstring;
{
    perror(errstring);
    exit(1);
}


/*******************************************************
****
The status is stored in a temp structure. If the driver is
the first one, it gets initialized and added in other cases

*******************************************************
***/

save_stat(instance)
int instance;
{
    int num;

    if ( instance == 0 ) {
    /* Master Driver */
        str_data.s_attached         = d_attached;
        str_data.s_transactions     = d_transactions;
        str_data.s_max_transactions = d_max_transactions;
        str_data.s_max_duration     = d_max_duration;

        for (num = 1; num < 6; num++) {
          str_data.s_stat[0].total_response[num] = d_stat.total_response[num];
          str_data.s_stat[0].bad[num]            = d_stat.bad[num];
          str_data.s_stat[0].count[num]          = d_stat.count[num];
        }
    } else {
        str_data.s_attached     += d_attached;
        str_data.s_transactions += d_transactions;

        for (num = 1; num < 6; num++) {
          str_data.s_stat[0].total_response[num] += d_stat.total_response[num];
          str_data.s_stat[0].bad[num]            += d_stat.bad[num];
          str_data.s_stat[0].count[num]          += d_stat.count[num];
        }
    }
}


/*******************************************************
****
updates the actual shared memory

*******************************************************
***/

update_shm()
{

    int num=0;

    m_attached         = str_data.s_attached;
    m_transactions     = str_data.s_transactions;
    m_max_transactions = str_data.s_max_transactions;
    m_max_duration     = str_data.s_max_duration;

    for (num = 1; num < 6; num++) {
```

```
364          shm_master->s_stat[0].total_response[num] =
365          str_data.s_stat[0].total_response[num];
366          shm_master->s_stat[0].bad[num] =
367          str_data.s_stat[0].bad[num];
368          shm_master->s_stat[0].count[num] =
369          str_data.s_stat[0].count[num];
370      }
371  }
372
373
 /*******************************************************
 **********
374  initialize_stuff takes care of all initialization
375
 *******************************************************
 *********/
376
377  initialize_stuff(argc, argv)
378  int argc;
379  char **argv;
380  {
381
382      extern struct timeval start_time;
383
384      /* create the shared memory */
385      shm_master = attach_shm(SHMKEY_MASTER,
 sizeof(shm_t),1);
386
387      /* Initialize to WAIT by default */
388
389      m_state = RUN;
390      m_desired = atoi(argv[2]);
391      m_max_transactions = 0x7fffffff;
392      m_max_duration = 9000.0;  /* 2.5 hours */
393
394
395      /* initialize the clock */
396
397      initclock();
398      shm_master->start_time = start_time;
399  }
400
401
 /*******************************************************
 *******
402  cleanup_shm detaches from the shared memory region
403  and makes the state=STOP in the slave shared memory
404
405  This is to take care of condition of stopping the pro-
 cess
406  from the external sources.
407
 *******************************************************
 ******/
408
409  cleanup_shm(n_drivers)
410  int n_drivers;
411  {
412  int ret,i;
413      for ( i=0; i < n_drivers; i++ )
414      {
415          d_desired = 0;
416          d_state = STOP;
417          ret =
 write(sd[i],&xfer_data,sizeof(xfer_data));
418          if (ret < 0 ) {
419              Error("Shared memory write");
420          fprintf(stderr,"Ouch!!! couldn't write to
 driver %d (2)\n",i);
421      }
422          close(sd[i]);
423      }
424
425      detach_shm((void *)shm_master);
426  }
```

driver/qualify.c

```
1
/*******************************************************
**********************
 2   @(#) Version: A.10.10 $Date: 97/02/27 23:27:06 $
 3
 4   (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*******************************************************
**********************/
 6  #include <unistd.h>
 7  #include <stdio.h>
 8  #include <values.h>
 9  #include <time.h>
10  #include "tpcc.h"
11
12
13  typedef struct
14      {
15      double min;
16      double max;
17      double min_so_far;
18      double max_so_far;
19      double total;
20      int bin_count;
21      int count;
22      int bin[2];
23      } bin_t;
24
25  bin_t *allocate_bin();
26  double median();
27  double mean();
28
29
30
31  /*****************************
32  Statistics to keep
33  *****************************/
34
35  /* statistics over entire run */
36  bin_t *throughput;      /* throughput throughout
 entire run */
37  double earliest;
38  double latest;
39  int total[7];           /* total transactions over
 entire run */
40
41  /* statistics for each transaction type */
42  bin_t *response[7];     /* response time distribution
 for each transaction */
43  bin_t *think[7];        /* think time for each trans-
 action */
44  bin_t *menu[7];         /* screen response time for
 each transaction */
45  bin_t *key[7];          /* key time for each transac-
 tion */
46  int count[7];           /* count of transactions
 within window */
47
48  /* statistics for delivery transactions */
49  bin_t *delivery_que;
50  bin_t *delivery_execution;
51  bin_t *delivery_response;
52
53  /* special statistics for individual transactions */
54  int ol_cnt;            /* total number of order lines
 */
55  int remote_ol_cnt;      /* total number of remote
 order lines */
56  int remote_payment;     /* number of remote payment
 transactions */
57  int remote_ordstat;     /* number of remote order
 status transactions */
58  int byname_payment;
59  int byname_ordstat;
60  int dist_skipped;  /* delivery districts skipped */
61  int d_skipped; /* deliver transactions with skipped
```

```
districts */
int all_local;          /* neworder transaction with all items local */
int no_skipped;       /* New Order with invalid items */
int fatal_count;              /* Other errors -- DISQUALIFIES RUN! */


/* configuration stuff */
TIME start, stop;
char *success;
char **result;
int n_result;

main(argc, argv)
    int argc;
    char **argv;
    {
    int i;

    /* initialize the data structures */
    setup(argc, argv);

    /* accumulate statisticss from the success file */
    accumulate_success_file(success);

    /* accumulate statistics from each result file */
    for (i=0; i<n_result; i++)
        accumulate_result_file(result[i]);

    cleanup();
    }


struct timeval start_time;

accumulate_success_file(name)
    char *name;
    {
    FILE *f;
    success_t s[1];
    success_header_t h[1];

    /* open the file */
    f = fopen(name, "r");
    if (f == NULL)
        error("Can't open success file %s\n", name);

    /* read the header */
    if (fread(h, sizeof(*h), 1, f) != 1)
       syserror("Can't read header from success file %s\n", name);
    start_time = h->start_time;

    while (read_success(f, s))
        count_success_record(s);

    fclose(f);
    }


int read_success(f, s)
    FILE *f;
    success_t *s;
    {
    char c;
    unsigned short len;

    /* read the length field first */
    if (fread(&len, sizeof(len), 1, f) != 1)
    return NO;

    /* read the success information */
    if (fread(s, sizeof(*s), 1, f) != 1)
    return NO;


    /* skip over the transaction stuff */
    while (len-- > sizeof(*s))
    if (fread(&c, 1, 1, f) != 1)
        return NO;
    return YES;
    }


accumulate_result_file(name)
    char *name;
    {
    delivery_trans t[1];
    FILE *f;

    /* open the file */
    f = fopen(name, "r");
    if (f == NULL)
        error("Can't open result file %s\n", name);

    while (fread(t, sizeof(*t), 1, f) == 1)
        count_result_record(t);

    fclose(f);
    }


setup(argc, argv)
    int argc;
    char **argv;
    {

    /* configure */
    configure(argc, argv);

    /* initialize the statistics */
    init_statistics();
    }


init_statistics()
    {
    int i;

    throughput = allocate_bin(0.0, 200.0, 2000);  /* Runs may be 200 minutes. */
    latest = MINDOUBLE;
    earliest = MAXDOUBLE;

    /* do for each transaction type */
    for (i=1; i<=DEFERRED; i++)
        {
        /* TPC-C spec says times must be reported accurate to .1 sec */
        menu[i] = allocate_bin(0.0, 10.0, 100);
        key[i] = allocate_bin(0.0, 30.0, 3000);
        response[i] = allocate_bin(0.0, 30.0, 3000);
        /* for 90th%ile to .01 */
        think[i] = allocate_bin(0.0, 200.0, 2000);
        total[i] = 0;
        count[i] = 0;
        }

    delivery_response = response[DEFERRED];
    delivery_que = menu[DEFERRED];
    delivery_execution = key[DEFERRED];

    ol_cnt = 0;
    remote_ol_cnt = 0;
    all_local = 0;
    remote_payment = 0;
    remote_ordstat = 0;
    byname_payment = 0;
    byname_ordstat = 0;
    dist_skipped = 0;
```

```
207      d_skipped = 0;
208
209      no_skipped = 0;
210      fatal_count = 0;
211      }
212
213
214  count_success_record(t)
215      success_t *t;
216      {
217      if (t->type <= 0 || t->type > 5)
218          error("Bad success record\n");
219
220      if (t->status != OK && !(t->type == NEWORDER && t-
 >status == E_INVALID_ITEM)
221  )       {
222          fatal_count++;
223          return;
224          }
225
226      /* count the number of "acceptable" transactions
 */
227      total[t->type]++;
228      if (t->type == NEWORDER)
229          addbin(throughput, t->t3 / 60);
230
231      /* keep track of earliest and latest transactions
 */
232      if (t->t3 < earliest) earliest = t->t3;
233      if (t->t4 > latest)   latest   = t->t4;
234
235      /* if outside window, then done */
236      if (t->t3 < start ||  t->t4 > stop)
237      return;
238
239      /* keep track of transactions within the window
 */
240      count[t->type]++;
241
242      /* accumulate distributions */
243      addbin(menu[t->type], t->t2 - t->t1);
244      addbin(key[t->type], t->t3 - t->t2);
245      addbin(response[t->type], t->t4 - t->t3);
246      addbin(think[t->type], t->t5 - t->t4);
247
248      /* count the misc stuff */
249      if (t->type == PAYMENT)   remote_payment += t-
 >remote;
250      if (t->type == PAYMENT)   byname_payment += t-
 >byname;
251      if (t->type == ORDSTAT)   remote_ordstat += t-
 >remote;
252      if (t->type == ORDSTAT)   byname_ordstat += t-
 >byname;
253      if (t->type == NEWORDER)  ol_cnt += t->ol_cnt;
254      if (t->type == NEWORDER)  remote_ol_cnt += t-
 >remote_ol_cnt;
255      if (t->type == NEWORDER)  all_local += (t-
 >remote_ol_cnt == 0);
256
257      /* count the number of errors */
258      if (t->type == NEWORDER && t->status ==
 E_INVALID_ITEM)
259          no_skipped++;
260      else if (t->status != OK)
261          fatal_count++;
262      }
263
264
265  count_result_record(t)
266      delivery_trans *t;
267      {
268      TIME enque, deque, complete;
269      int d, skipped;
270
271      /* convert the times */
272      enque = elapsed_time(t->enque);
273      deque = elapsed_time(t->deque);
274      complete = elapsed_time(t->complete);
275
276      total[DEFERRED]++;
277
278      /* keep track of earliest and latest transactions
 */
279      if (enque < earliest) earliest = enque;
280      if (complete > latest)latest   = complete;
281
282      /* if outside window, then done */
283      if (complete > stop || enque < start) return;
284
285      /* keep track of transactions within the window
 */
286      count[DEFERRED]++;
287      addbin(delivery_response, complete - enque);
288      addbin(delivery_que, deque - enque);
289      addbin(delivery_execution, complete - deque);
290
291      /* for each district, check for errors and
 skipped districts */
292      skipped = 0;
293      for (d=0; d<10; d++)
294      {
295      if (t->order[d].status == E_NOT_ENOUGH_ORDERS)
 skipped++;
296      else if (t->order[d].status != OK)
 fatal_count++;
297      }
298
299      /* accumulate info about skipped districts */
300      dist_skipped += skipped;
301      d_skipped += (skipped != 0);
302      }
303
304
305  display_statistics()
306      {
307      int count_all;
308      int i;
309      int count_menu;
310      double total_menu;
311
312      /* Adjust the window if actual window was smaller
 */
313      if (start < earliest)   start = earliest;
314      if (stop > latest)      stop = latest;
315
316      /* display the start date */
317      printf("                    %s\n",
 ctime(&start_time.tv_sec));
318
319      /* display the overall results */
320      printf("MQTH, Compute Maximum Qualified Through-
 put     %20.2f tpmC\n",
321           60 * count[NEWORDER] / (stop-start) );
322
323      /* Display response times */
324      printf("\n");
325      printf("Response Times (90th percintile/Aver-
 age/maximum) in seconds\n");
326      for (i=NEWORDER; i<= DEFERRED; i++)
327      printf("- %-48s %6.2f /%4.2f /%5.2f\n",
 transaction_name[i],
328          median(response[i], .9),   mean(response[i]),
329          response[i]->max_so_far);
330
331      /* display response time delays */
332      /* later -- add to success header */
333
334      /* Transaction Mix */
335      /* Calculate how many transactions total occurred
 in window */
336      count_all = 0;
337      for (i=NEWORDER; i < DEFERRED; i++)
338      count_all += count[i];
```

```
339
340      /* display the mix */
341      printf("\n");
342      printf("Transaction Mix, in percent of total
 transactions\n");
343      for (i=NEWORDER; i<DEFERRED; i++)
344          if (count[i] > 0)
345          percent(transaction_name[i], count[i],
 count_all);
346
347      /* Menu response times */
348      printf("\n");
349      printf("Menu Response Times (in seconds),    "
350             "Min         Average      Max\n");
351      for (i = NEWORDER;  i<DEFERRED;  i++)
352      printf("- %-30s  %4.2f           %4.2f
 %4.2f\n",
353           transaction_name[i],
354           menu[i]->min_so_far,
355           mean(menu[i]),
356           menu[i]->max_so_far);
357
358      /* Display the aggegate response times */
359      total_menu = 0;  count_menu = 0;
360      for (i = NEWORDER;  i<DEFERRED; i++)
361         {total_menu += menu[i]->total;   count_menu +=
 menu[i]->count;}
362     printf("- %-30s            %4.2f\n","All Menus",
 total_menu/count_menu);
363
364      /* Key, Think Times */
365      printf("\n");
366      printf("Keying/Think Times (in seconds),     "
367         "Min            Average         Max\n");
368      for (i = NEWORDER;  i<DEFERRED;  i++)
369      printf("- %-28s  %7.2f /%5.2f    %5.2f /%5.2f
 %6.2f /%6.2f\n",
370           transaction_name[i],
371           key[i]->min_so_far,  think[i]->min_so_far,
372           mean(key[i]), mean(think[i]),
373           key[i]->max_so_far, think[i]->max_so_far);
374
375      /* Test Duration */
376      printf("\n");
377      printf("Test Duration\n");
378     printf("- Ramp-up time                    %15.2f
 minutes\n", start/60);
379      printf("- Measurement interval
 %15.2f minutes\n",
380            (stop-start)/60);
381      printf("- Number of transactions (all types)
 %23d\n", count_all);
382      printf("  completed in measurement interval\n");
383
384      /* Other stuff */
385      printf("\n");
386      printf("Other numerical quantities required in
 Full Disclosure Report\n");
387      if (fatal_count != 0)
388          printf("- Fatal errors             %20d\n",
 fatal_count);
389      percent("New Orders with bad
 items:",no_skipped,count[NEWORDER]);
390      value  ("Order lines per order:", (double)ol_cnt
 / count[NEWORDER]);
391      percent("Remote order lines:", remote_ol_cnt,
 ol_cnt);
392      percent("Orders with all local items:",
 all_local, count[NEWORDER]);
393      percent("Remote payment:", remote_payment,
 count[PAYMENT]);
394      percent("Payment by last name:", byname_payment,
 count[PAYMENT]);
395      percent("Order status by last name:",
 byname_ordstat, count[ORDSTAT]);
396      percent("Deliveries with skipped dis-
 tricts:",d_skipped,count[DEFERRED]);
397
398      /* Graphs */
399      printf("
400  \n");
401
 printf("*********************************************
 **********\n");
402      /* show the throughput graph */
403      printf("\nThroughput\n");
404      display_bin(throughput, 1.0);
405
406      /* Show the graphs for each transaction */
407      for (i=1; i<DEFERRED; i++)
408          {
409
410          printf("\nMenu response times for %s\n",
 transaction_name[i]);
411          display_bin(menu[i], 1.0 / menu[i]->count);
412
413          printf("\nKey times for %s\n",
 transaction_name[i]);
414          display_bin(key[i], 1.0 / key[i]->count);
415
416          printf("\nResponse times for %s\n",
 transaction_name[i]);
417          display_bin(response[i], 1.0 / response[i]-
 >count);
418
419          printf("\nThink times for %s\n",
 transaction_name[i]);
420          display_bin(think[i], 1.0 / think[i]->count);
421          }
422
423      printf("\nQue times for %s\n",
 transaction_name[DEFERRED]);
424      display_bin(delivery_que, 1.0 / delivery_que-
 >count);
425
426      printf("\nExecute times for %s\n",
 transaction_name[DEFERRED]);
427      display_bin(delivery_execution, 1.0 /
 delivery_execution->count);
428
429      printf("\nResponse times for %s\n",
 transaction_name[DEFERRED]);
430      display_bin(delivery_response, 1.0 /
 delivery_response->count);
431
432      }
433
434
435
436  configure(argc, argv)
437
 /*****************************************************
 ************
438  configure_test processes the command string and ini-
 tializes the overall test
439
 *****************************************************
 ***************/
440      int argc;
441      char **argv;
442      {
443      double atof();
444      extern char *optarg;
445      extern int optind, opterr;
446      char ch;
447      int i;
448
449      /* define the default configuration */
450      start = 0.0;
451      stop =  120*60;
452
453      /* while there are options */
454      while ((ch = getopt (argc, argv, "s:S:")) != -1)
455
```

```
456          /* process according to options */
457          switch ( ch )
458              {
459              case 's':  start = atof(optarg)*60.0;
460                         break;
461
462              case 'S':  stop = atof(optarg)*60.0;
463                         break;
464
465              default:   error("Bad runstring argu-
 ment.\n");
466                         break;
467              }
468
469      /* error if any options left over */
470      if (optind == argc)
471          error("qualify -s <start> -S <stop> <success
 file> <result files>\n");
472
473      success = argv[optind];
474      result  = &argv[optind+1];
475      n_result = argc - optind - 1;
476      }
477
478
479
480
481
482  cleanup()
483      {
484      display_statistics();
485      }
486
487
488
489  percent(str, partial, full)
490      char *str;
491      int partial;
492      int full;
493      {
494      double p;
495
496      if (full == 0)   p = 0;
497      else             p = 100.0 * partial / full;
498
499      printf("- %-40s %6.2f%%  (%d of %d)\n", str, p,
 partial, full);
500      }
501
502
503  value(str, val)
504      char *str;
505      double val;
506      {
507      printf("- %-40s %6.2f\n", str, val);
508      }
509
510
511
512
513  bin_t *allocate_bin(min, max, bin_count)
514
 /*******************************************************
 ***********
515  allocate_bin creates and initializes a bin structure
516
 *******************************************************
 **********/
517      double min;
518      double max;
519      int bin_count;
520      {
521      int i;
522      bin_t *bin;
523
524      /* allocate the memory for the bin */
525      bin = malloc(sizeof(bin_t) +
 (bin_count+2)*sizeof(bin->bin[0]));
526      if (bin == NULL)
527          error("allocate_bin: couldn't allocate mem-
 ory. count=%d\n", bin_count);
528
529      /* initialize the structures */
530      bin->min = min;
531      bin->max = max;
532      bin->bin_count = bin_count;
533      bin->min_so_far = MAXDOUBLE;
534      bin->max_so_far = MINDOUBLE;
535      bin->count = 0;
536      bin->total = 0.0;
537      for (i=0; i<bin_count+2; i++)
538          bin->bin[i] = 0;
539
540      return bin;
541      }
542
543
544  display_bin(bin, yscale)
545      bin_t *bin;
546      double yscale;
547      {
548      int first, last, i;
549      double xscale;
550
551      /* show the statistics  */
552      printf("  count=%d  min=%g  max=%g  90%%=%g  aver-
 age=%g\n",
553          bin->count, bin->min_so_far, bin->max_so_far,
554          median(bin, .9), bin->total/bin->count);
555
556      /* dump the graph data */
557      printf("  Distribution:\n");
558      xscale = (bin->max - bin->min) / bin->bin_count;
559      first = mapbin(bin, bin->min_so_far);
560      last  = mapbin(bin, bin->max_so_far);
561      for (i=first; i <= last; i++)
562          printf("   %10g %10g\n",   i*xscale, bin-
 >bin[i] / xscale * yscale);
563      }
564
565
566
567  addbin(bin, value)
568
 /*******************************************************
 ******************
569  addbin adds a new value to a bin
570
 *******************************************************
 *****************/
571      bin_t *bin;
572      double value;
573      {
574
575      /* add in the appropriate values */
576      bin->bin[mapbin(bin, value)]++;
577      bin->count++;
578      bin->total += value;
579      if (value < bin->min_so_far)  bin->min_so_far =
 value;
580      if (value > bin->max_so_far)  bin->max_so_far =
 value;
581      }
582
583  double mean(bin)
584      bin_t *bin;
585      {
586      return bin->total / bin->count;
587      }
588
589  double median(bin, percentile)
590      bin_t *bin;
591      double percentile;
592      {
```

```
593      int i;
594      int count;
595      double value;
596      int total = bin->count;        /* total number of
  events in bins */
597      int maxbin = bin->bin_count+2;  /* number of bins
  */
598
599      /* scan the bins until the fraction is reached */
600      count = 0;
601      for (i=0; i<=maxbin; i++)
602          {
603          count += bin->bin[i];
604          if (count >= percentile * total) break;
605          }
606
607      if (i <= maxbin)
608      value = bin->min + i * (bin->max - bin->min) /
  bin->bin_count;
609      else
610      value = MAXDOUBLE;
611
612      return value;
613      }
614
615
616
617
618
619
620
621  int mapbin(bin, value)
622      bin_t *bin;
623      double value;
624      {
625      int i;
626      if (value < bin->min)  i = 0;
627      else if (value > bin->max)  i = bin->bin_count +
  1;
628      else
629         i = (value - bin->min) / (bin->max - bin->min)
  * bin->bin_count + 1;
630      return i;
631      }
632
```

driver/slave.c

```
  1
  /*******************************************************
  **********************
   2  @(#) Version: A.10.10 $Date: 97/02/27 23:27:07 $
   3
   4  (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
   5
  *******************************************************
  *********************/
   6
   7  #include <sys/types.h>
   8  #include <sys/socket.h>
   9  #include <sys/ioctl.h>
  10  #include <sys/ipc.h>
  11  #include <netinet/in.h>
  12  #include <netdb.h>
  13  #include <stdio.h>
  14  #include <pwd.h>
  15  #include "master.h"
  16  #include "shm.h"
  17
  18  #define TKJOS_DEBUG
  19  #ifdef TKJOS_DEBUG
  20  #define LOG(X) { \
  21      FILE *x; \
  22      x=fopen("/tmp/slave.log","a"); \
  23      fprintf X ;\
  24      fclose(x); }
  25  #else
  26  #define LOG(X)
  27  #endif
  28
  29
  30
  31
  32  int main(argc, argv)
  33
  34
  /*******************************************************
  **********
  35  TPCC information sharing program.(slave)
  36  This gets spawned from the master process.
  37
  *******************************************************
  *********/
  38
  39  int argc;
  40  char **argv;
  41  {
  42
  43
  44
  45      /* create the shared memory */
  46      /* initialize everything */
  47      initialize_stuff(argc, argv);
  48
  49      update_info();
  50
  51      cleanup_shm();
  52  }
  53
  54  void
  55  send_receive_data()
  56  {
  57      int ret;
  58
  59      ret = read(0,&xfer_data,sizeof(xfer_data));
  60      if (ret < 0 ) {
  61      LOG((x,"Ouch! Couldn't read from master\n"));
  62              Error("Shared memory read");
  63      }
  64
  65      state = d_state;
  66      desired = d_desired;
  67      shm->start_time = d_start_time;
  68
  69      d_attached = attached;
  70      d_transactions = transactions;
  71      d_max_transactions = max_transactions;
  72      d_max_duration = max_duration;
  73      d_stat = *stat;
  74
  75
  76      ret = write(1,&xfer_data,sizeof(xfer_data));
  77      if (ret < 0 ) {
  78      LOG((x,"Ouch! Couldn't write to master\n"));
  79          Error("Shared memory write");
  80      }
  81  }
  82
  83
  /*******************************************************
  **********
  84  Read the information from the master and return the
  status
  85  till the state = STOP.
  86
  *******************************************************
  *********/
  87
  88  update_info()
  89  {
  90  int ret;
  91
  92      if ( state != WAIT )
```

```
 93      {
 94      /*
 95       * The driver process is supposed to start us off
 by setting the state
 96       * to WAIT.  But if there is too many of them
 spawning, it's possible
 97       * none has had the chance to clear the "state"
 variable from the STOP
 98       * setting at the end of the last run.  Or, the
 state may be
 99       * left at RUN from a previous run.  Force state
 to WAIT.
100       */
101       state = WAIT;
102      }
103
104
105      while ( state != STOP || (state == STOP &&
 attached != 0))
106      {
107      send_receive_data();
108      }
109     /* one last time to make sure everything was sent.
 This is needed
110        because attached may go to zero after doing the
 last write, thus
111        if you don't send this info, the master may not
 know that everyone
112         has "detached" from the system */
113      send_receive_data();
114  }
115
116  Error(errstring)
117  char *errstring;
118  {
119  extern int errno;
120  LOG((x,"Error=%d\n",errno));
121      perror(errstring);
122      exit(1);
123  }
124
125
 /*********************************************************
 **********
126  initialize_stuff takes care of all initialization
127
 *********************************************************
 *********/
128
129  initialize_stuff(argc, argv)
130  int argc;
131  char **argv;
132  {
133      char *eptr;
134      key_t key = 0;
135
136      /* get the shared memory key.
137       *
138       * Take it from the command line if specified.  If
 not
139       * check the environment.  If not then use the
 hardcoded value.
140       */
141
142      if (argc > 1)
143          key = atoi(argv[1]);
144
145      if (key == 0) {
146          eptr = getenv("TPCC_SHMKEY_DRIVER");
147          if (eptr == NULL) key = SHMKEY_DRIVER;
148          else              key = atoi(eptr);
149      }
150
151      /* create the shared memory */
152      shm = attach_shm(key, sizeof(shm_t),0);
153      if (shm == NULL) {
154          Error("Slave:  Could not attach to shared mem-
 ory (key = %d)",
155                  key);
156      }
157  }
158
159
 /*********************************************************
 *******
160  cleanup_shm detaches from the shared memory region
161  and makes the local shm state=STOP.
162
 *********************************************************
 ******/
163
164  cleanup_shm()
165  {
166  int ret;
167
168      ret = read(0,&xfer_data,sizeof(xfer_data));
169      if (ret < 0 ) {
170      LOG((x,"Ouch! Couldn't read to master (1)\n"));
171          Error("Shared memory read");
172      }
173      state = d_state;
174      desired = d_desired;
175
176      detach_shm((void *)shm);
177  }
```

driver/socket.c

```
  1
 /*********************************************************
 **********************
  2   @(#) Version: A.10.10 $Date: 97/02/27 23:27:07 $
  3
  4   (c) Copyright 1996, Hewlett-Packard Company, all
 rights reserved.
  5
 *********************************************************
 **********************/
  6
  7  #include <stdlib.h>
  8  #include <sys/types.h>
  9  #include <sys/socket.h>
 10  #include <netinet/in.h>
 11  #include <netdb.h>
 12  #include <ctype.h>
 13  #include <stdio.h>
 14  #include <signal.h>
 15  #include <errno.h>
 16  extern int errno;
 17  #include <netinet/in.h>
 18  #include <netinet/tcp.h>
 19  #include <netdb.h>
 20
 21  int fd;
 22  extern int clientnum;
 23
 24  connection_begin(userid, clnt)
 25      int userid, clnt;
 26  {
 27      int fd = connect_to_server(userid, clnt);
 28
 29      /* adopt the connection as stdin and stdout */
 30      fflush(stdout);
 31      dup2(fd, 0);
 32      dup2(fd, 1);
 33      close(fd);
 34  }
 35
 36  connect_to_server(userid, clnt)
 37      int userid, clnt;
 38      {
 39      char *sut;
 40      char *service;
 41      char *str;
```

```
42      int local_port=0;
43      char buf[120], buf2[120];
44
45      /* Decide which SUT to use */
46      sut = getenv("DRIVER_SUT");
47      if (!sut)
48      {
49          sut = getenv("CLIENT");
50          if (sut == NULL)
51          sut = "client";
52
53          /* If using multiple suts, append number to
 sut name */
54          str = getenv("NR_CLIENT");
55          /* Here it seems if you set NR_CLIENT to 1,
 you'll have to name */
56          /* your client "xxx1" even though there is
 only one of them.    */
57          str = getenv("NR_LAN");
58          if (str != NULL)
59          {
60          /* If there are multiple lans to clients,
 access them as */
61         /* "clientX_Y"  where X is client # and Y the
 lan #.    */
62             sprintf(buf, "%s%d_%d", sut, clnt, (userid
 % atoi(str)) + 1);
63              sut = buf;
64          }
65          else
66          {
67          /* Client number is calculated in user()
 before we get here.*/
68              sprintf(buf, "%s%d", sut, clnt);
69              sut = buf;
70          }
71      }
72
73      service = getenv("TPCC_SERVICE");
74      if (service == NULL) service = "tpcc";
75
76      /* yes, they both should be '=' */
77     if ((str = getenv("LOCAL_PORT")) && (local_port =
 atoi(str)))
78      local_port += userid;
79
80      /* connect to the server */
81      fd = connect_server(sut, service, local_port);
82      if (fd < 0)
83          syserror("Can't connect to '%s' on machine
 '%s' \n", service, sut);
84
85      return fd;
86      }
87
88
89  void connection_done()
90      {
91      fflush(stdout);
92      close(1); close(0);
93      }
94
95
96  int connect_server(servername, service, localport)
97
 /*********************************************************
 ********************
98  connect_server connects to the desired tcp service
99
 *********************************************************
 ******************/
100     char *servername;
101     char *service;
102     int localport;
103     {
104     int fd;
105     struct sockaddr_in address;
106     struct hostent *host;
107     struct servent *server;
108     static struct linger no_linger= {1,0};
109     int yes = 1;
110     int port;
111     char *s;
112
113     /* get the host address */
114     host = gethostbyname(servername);
115     if (host == NULL) return -1;
116
117    /* if the service is all digits, then use that as
 tcp port number */
118     for (s=service; isdigit(*s); s++)
119     ;
120     if (*s == '\0')
121     port = atoi(service);
122
123     /* otherwise, get the named service port number
 */
124     else
125     {
126         server = getservbyname(service, "tcp");
127         if (server == NULL)
128         error("Service %s is unknown\n", service);
129         port = server->s_port;
130     }
131
132     /* create a socket */
133     fd = socket(host->h_addrtype, SOCK_STREAM, 0);
134     /* if (fd < 0)  goto error; */
135     if (fd < 0)
136     {
137        syserror("Can't create socket.  servername=%s
 service=%s\n",
138                                             servername,
 service);
139     goto error;
140     }
141
142     if (prepare_socket(fd) < 0)
143     message("prepare_socket failed\n");
144
145     /* build but the source address */
146     memset(address, 0, sizeof(address));
147     address.sin_family = AF_INET;
148     address.sin_port = localport;
149     address.sin_addr.s_addr = INADDR_ANY;
150     if (bind(fd, &address, sizeof(address)) == -1)
151     {
152             message("**WARNING** Unable bind local
 port %d, errno %d\n", localport, errno);
153          address.sin_port = 0;
154            if (bind(fd, &address, sizeof(address)) ==
 -1)
155          {
156          message("\tsubsequent rebind to 0 failed
 also\n");
157          }
158     }
159
160
161     /* reuse the address structure, building the des-
 tination address */
162     memset(address, 0, sizeof(address));
163     /* build up an internet style address for the
 host*/
164     address.sin_family = host->h_addrtype;
165     address.sin_port = port;
166     memcpy(&address.sin_addr, host->h_addr, host-
 >h_length);
167
168     /* connect the socket to the remote port */
169     while (connect(fd, &address, sizeof(address)) <
 0)
170     {
171         if (errno == ETIMEDOUT)
```

```
172              delay(.5);
173          else
174      if (errno == EADDRINUSE)
175      {
176          close(fd);
177            fd = socket(host->h_addrtype, SOCK_STREAM,
 0);
178      }
179      {
180              syserror("Can't connect socket.  server-
 name=%s  service=%s, port=%d localport=%d\n", server-
 name, service, port, localport);
181              goto error;
182      }
183      }
184
185      return fd;
186
187  error:
188      close(fd);
189      return -1;
190      }
191
192
193
194  int prepare_socket(fd)
195      int fd;
196      {
197      int yes = 1;
198
199      /* if (setsockopt(fd, SOL_SOCKET, SO_KEEPALIVE,
 &yes, sizeof(yes)) < 0)
200          return -1; */
201      if (setsockopt(fd, SOL_SOCKET, SO_REUSEADDR,
 &yes, sizeof(yes)) < 0)
202          return -1;
203     if (setsockopt(fd, SOL_SOCKET, TCP_NODELAY, &yes,
 sizeof(yes)) < 0)
204      return -1;
205      /* SO_SNDBUF, SO_RCVBUF, TPC_MAXSEG  should be
 set ?? */
206      return 0;
207      }
208
```

lib/date.c

```
  1
  /*****************************************************
  **********************
   2  @(#) Version: A.10.10 $Date: 96/04/02 16:26:09 $
   3
   4  (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
   5
  *****************************************************
  **********************/
   6  #include "tpcc.h"
   7  #include <time.h>
   8
   9  /* macro to get starting day of a particular year
  (1901 thru 2100) */
  10  #define YEAR(yr)  ( (yr-1900)*365 + (yr-1900-1)/4 )
  11
  12  CurrentDate(date)
  13
  /*****************************************************
  **************
  14  CurrentDate fetches the current date and time
  15
  *****************************************************
  **************/
  16      DATE *date;
  17      {
  18      struct timeval time;
  19      struct timezone tz;
  20
  21      /* get the current time of day */
  22      if (gettimeofday(&time, &tz) < 0)
  23      syserror("Can't get time of day\n");
  24
  25      /* adjust the time of day by the timezone */
  26      time.tv_sec -= tz.tz_minuteswest * 60;
  27
  28      /* convert seconds and days since EPOCH (Jan 1,
 1970) */
  29      date->day = time.tv_sec / (24*60*60);
  30      date->sec = time.tv_sec - date->day * (24*60*60);
  31
  32      /* convert to days since Jan 1, 1900 */
  33      date->day += YEAR(1970);
  34      }
  35
  36
  37  EmptyDate(date)
  38  /*****************************************************
  39  Get a NULL date and time
  40
  *****************************************************/
  41      DATE *date;
  42      {
  43      date->day = 0;   /* Use EMPTYNUM instead */
  44      date->sec = 0;
  45      }
  46
  47  int IsEmptyDate(date)
  48      DATE *date;
  49      {
  50      return (date->day == 0 & date->sec == 0);
  51      }
  52
  53
  54  #define Feb29 (31+29-1)
  55
  56  fmt_date(str, date)
  57
  /*****************************************************
  ***************
  58  fmt_date formats the DATE into a string  MM-DD-YY HH-
 MM-SS
  59
  *****************************************************
  *************/
  60      char str[20];
  61      DATE *date;
  62      {
  63      /* Note: should probably do date and time sepa-
 rately */
  64
  65      int quad, year, month, day;
  66      int hour, minute, sec;
  67
  68      static int dur[] = {31, 28, 31, 30, 31, 30, 31,
 31, 30, 31, 30, 31};
  69      static int first = YES;
  70
  71      day = date->day;
  72      sec = date->sec;
  73
  74      /* if NULL date, then return empty string */
  75      if (day == EMPTY_NUM || sec == EMPTY_NUM)
  76      {str[0] = '\0';  return;}
  77
  78      /* 2100, 1900 are NOT leap years.  If we are Feb
 29 or later, add a day */
  79      if (day >= Feb29 + YEAR(2100))  day++;
  80      if (day >= Feb29)               day++;
  81
  82      /* figure out which quad and day within quad we
 are in */
  83      quad = day / (4*365+1);
  84      day = day - quad * (4*365+1);
  85
  86      /* get our year within quad and day within the
```

```
year */
87      if      (day < 1*365+1)     {year = 0;}
88      else if (day < 2*365+1)     {year = 1; day -=
1*365+1;}
89      else if (day < 3*365+1)     {year = 2; day -=
2*365+1;}
90      else                        {year = 3; day -=
3*365+1;}
91
92      /* if this is a leap year, february has 29 days */
93      if (year == 0)          dur[1] = 29;
94      else                    dur[1] = 28;
95
96      /* decide which day and month we are */
97      for (month = 0; day >= dur[month]; month++)
98          day -= dur[month];
99
100     /* decide what time of day it is */
101     minute = sec / 60;
102     sec = sec - minute * 60;
103     hour = minute / 60;
104     minute = minute - hour * 60;
105
106     /* format the date and time */
107     fmtint(str+0,  day+1, 2, ' ');
108     str[2]='-';
109     fmtint(str+3, month+1, 2, '0');
110     str[5]='-';
111     fmtint(str+6, 1900+quad*4+year, 4, '0');
112     str[10] = ' ';
113     fmtint(str+11, hour, 2, ' ');
114     str[13] = ':';
115     fmtint(str+14, minute, 2, '0');
116     str[16] = ':';
117     fmtint(str+17, sec, 2, '0');
118     str[19] = '\0';
119     }
```

lib/delay.c

```
1
/*********************************************************
**********************
 2  @(#) Version: A.10.10 $Date: 97/02/14 12:31:59 $
 3
 4  (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*********************************************************
*********************/
 6 #include <sys/time.h>
 7 #include <sys/errno.h>
 8 #ifndef HPUX9
 9 #include <time.h>
10 #endif
11 #include "tpcc.h"
12 #include "shm.h"
13
14 delay(sec)
15
/*********************************************************
*****************
16 delay sleeps for the specified number of seconds.  (to
closest 1/100'th second)
17
*********************************************************
********************/
18      double sec;
19      {
20 #ifdef HPUX9
21      struct timeval delay;
22 #else
23      struct timespec delay;
24 #endif
25
26      /* if no delay,  done */
27      if (sec <= 0.0) return;
28
29      /* add a portion of a clock tick to keep averages
correct */
30      sec += 1.0 / CLK_TCK;
31
32      /* convert the delay to seconds and nanoseconds */
33      delay.tv_sec = sec;
34 #ifdef HPUX9
35      delay.tv_usec = (sec - delay.tv_sec) * 1000000;
36 #else
37     delay.tv_nsec = (sec - delay.tv_sec) * 1000000000;
38 #endif
39
40      /* sleep on a select call */
41 #ifdef HPUX9
42      if (select(0, NULL, NULL, NULL, &delay) < 0) {
43          syserror("delay:  select() call failed\n");
44      }
45 #else
46      if (nanosleep(&delay,NULL) == -1) {
47      if (errno != EINTR) {
48              syserror("delay:  nanosleep() call failed,
errno = %d\n",errno);
49      }
50      }
51 #endif
52 }
53
54
55
56 struct timeval start_time;
57
58 initclock()
59      {
60      gettimeofday(&start_time, NULL);
61      }
62
63
64 TIME getclock()
65
/*********************************************************
************
66 getclock returns the current time, expressed in sec-
onds from start of run
67
*********************************************************
*****************/
68      {
69      struct timeval current;
70      gettimeofday(&current, NULL);
71
72      return elapsed_time(&current);
73      }
```

lib/errlog.c

```
1
/*********************************************************
**********************
 2  @(#) Version: A.10.10 $Date: 96/06/11 10:46:41 $
 3
 4  (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*********************************************************
*********************/
 6 #include    <fcntl.h>
 7 #include    <stdio.h>
 8 #include    <unistd.h>
 9 #include    <errno.h>
10
11 #include <stdarg.h>
12 #include <stdlib.h>
13 #include <stdio.h>
14 #include <errno.h>
15 extern int errno;
16
```

```
int userid;


error(format, args)
/*******************************************************
**************
error formats a message and outputs it to a standard
location (stderr for now)
*******************************************************
****************/
    char *format;
    int args;
    {
    va_list argptr;

    /* point to the list of arguments */
    va_start(argptr, args);

    /* format and print to stderr */
    vmessage(format, argptr);

    /* done */
    va_end(argptr);

    /* take an error exit */
    exit(1);
    }


syserror( format, args )
/*******************************************************
**************
syserror logs a message with the system error code
*******************************************************
****************/
    char *format;
    int args;
    {
    va_list argptr;
    int save_errno = errno;

    /* point to the list of arguments */
    va_start(argptr, args);

    /* format and print to stderr */
    vmessage(format, argptr);

    /* done */
    va_end(argptr);

    /* display the system error message */
    message("    System error message: %s\n", strer-
ror(save_errno));

    /* take an error exit */
    exit(1);
    }



message(format, args)
/*******************************************************
**************
message formats a message and outputs it to a standard
location (stderr for now)
*******************************************************
****************/
    char *format;
    int args;
    {
    va_list argptr;

    /* point to the list of arguments */
    va_start(argptr, args);

    /* format and print to stderr */
    vmessage(format, argptr);

    /* done */
    va_end(argptr);
    }



vmessage(format, argptr)
/*************************************************

*************************************************/
    char *format;
    va_list argptr;
{
  char buf[3*1024];

  /* format a message id */
  sprintf(buf, "User %-6d Pid %-6d ", userid, get-
pid());

  /* format the string and print it */
  vsprintf(buf+strlen(buf), format, argptr);
  if (getenv("NO_ERROR_LOG") == NULL)
      msg_buf(buf, strlen(buf));
  if (getenv("NO_STDERR") == NULL)
      write(2, buf, strlen(buf));
}



static msg_buf(buf, size)
  char *buf;
  int size;
  {
  int   fd;
  char  *fname;

  /* get the file name to use */
  fname = getenv("ERROR_LOG");
  if (fname == NULL)
    fname = "/tmp/ERROR_LOG";

  /* get exclusive access to the error log file */
  fd= open(fname, O_WRONLY | O_CREAT, 0666);
  if (fd < 0)
    console_error("Can't open tpc error log file
'ERROR_LOG'\n");
  lockf(fd, F_LOCK, 0);

  /* write the new text at the end of the file */
  lseek(fd, 0, SEEK_END);
  write(fd, buf, size);

  /* release the file */
  /* fsync(fd); */
  lockf(fd, F_ULOCK, 0);
  close(fd);
  }



console_error(str)
   char *str;
   {
   int fd = open("/dev/tty", O_WRONLY);
   write(fd, str, strlen(str));
```

```
  154     close(fd);
  155     exit(1);
  156     }
```

lib/fmt.c

```
    1
    /*****************************************************
    **********************
    2   @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $
    3
    4   (c) Copyright 1996, Hewlett-Packard Company, all
   rights reserved.
    5
    *****************************************************
    **********************/
    6  #include "tpcc.h"
    7  #include "iobuf.h"
    8  #include <math.h>    /* needed for ceil (VM) */
    9  #include <strings.h>
   10
   11  /* formatting routines.  */
   12
   13  /* Note: Currently use integer routines to format and
    convert.  Need to
   14     modify the code for cases when integers don't
    work.  */
   15
   16  fmt_money(str, m, width)
   17      char *str;
   18      MONEY m;
   19      int width;
   20      {
   21
   22      if (m == EMPTY_FLT)
   23          {
   24          memset(str, '_', width);
   25          str[width] = '\0';
   26          return;
   27          }
   28
   29      /* format it as a number with a leading blank */
   30      *str = ' ';
   31      fmt_flt(str+1, m/100, width-1, 2);
   32
   33      /* fill in a leading dollar */
   34      while (*(str+1) == ' ')
   35          str++;
   36      *str = '$';
   37      }
   38
   39
   40  double cvt_money(str)
   41      char *str;
   42      {
   43      char temp[81], *t, *s;
   44      double cvt_flt(), f;
   45
   46      /* skip leading and trailing blanks */
   47      cvt_text(str, temp);
   48
   49      /* remove leading $ */
   50      if (*temp == '$')   t = temp + 1;
   51      else                t = temp;
   52
   53      /* start scan at current character */
   54      s = t;
   55
   56      /* allow leading minus sign */
   57      if (*s == '-')
   58      s++;
   59
   60      /* allow leading digits */
   61      while (isdigit(*s))
   62      s++;
   63
   64      /* allow decimal pt and two decimal digits */
   65      if (*s == '.')    s++;
   66      if (isdigit(*s))  s++;
   67      if (isdigit(*s))  s++;
   68
   69      /* There should be no more characters */
   70      if (*s != '\0') return INVALID_FLT;
   71
   72      /* convert the floating pt number */
   73      f = cvt_flt(t);
   74      if (f == EMPTY_FLT)        return EMPTY_FLT;
   75      else if (f == INVALID_FLT) return INVALID_FLT;
   76      else                       return rint(f*100);
   77      }
   78
   79
   80  fmt_num(str, n, width)
   81      char str[];
   82      int n;
   83      int width;
   84      {
   85      /* mark the end of the string */
   86      str[width] = '\0';
   87
   88      /* if empty number, return the empty field */
   89      if (n == EMPTY_NUM)
   90          memset(str, '_', width);
   91
   92      /* otherwise, convert the integer */
   93      else
   94          fmtint(str, n, width, ' ');
   95
   96      debug("fmt_num:  n=%d  str=%s\n", n, str);
   97      }
   98
   99
  100
  101  cvt_num(str)
  102      char str[];
  103      {
  104      char text[81];
  105      cvt_text(str, text);
  106      if (*text == '\0')
  107          return EMPTY_NUM;
  108      else
  109          return cvtint(text);
  110      }
  111
  112
  113
  114  fmt_flt(str, x, width, dec)
  115
   /*****************************************************
   ******************
  116  fmt_flt converts a floating pt number to a string
   "999999.9999"
  117
   *****************************************************
   ****************/
  118      char *str;
  119      double x;
  120      int width;
  121      int dec;
  122      {
  123      int negative;
  124      int integer, fract;
  125      double absolute;
  126
  127      static double pow10[] =
  128     {1., 10., 100., 1000., 10000., 100000., 1000000.,
   10000000., 100000000.};
  129
  130      /* mark the end of string */
  131      str[width] = '\0';
  132
  133      /* if empty value, make it be an empty field */
  134      if (x == EMPTY_FLT)
  135          {
```

```
136         memset(str, '_', width);
137         return;
138         }
139
140     absolute = (x < 0)? -x: x;
141
142     /* separate into integer and fractional parts */
143     integer = (int) absolute;
144     fract   = (absolute - integer) * pow10[dec] + .5;
145
146     /* let the integer portion contain the sign */
147     if (x < 0) integer = -integer;
148
149     /* Format integer and fraction separately */
150     fmtint(str, integer, width-dec-1, ' ');
151     str[width-dec-1] = '.';
152     fmtint(str+width-dec, fract, dec, '0');
153     }
154
155
156
157
158 double cvt_flt(str)
159     char str[];
160     {
161     char text[81];
162     char *t;
163     double value;
164     int div;
165     int fract;
166     int negative;
167     int i;
168
169     /* normalize the text */
170     cvt_text(str, text);
171     if (*text == '\0')
172         return EMPTY_FLT;
173
174     negative = NO;
175     fract = NO;
176     value = 0;
177     div = 1.0;
178
179     negative = (text[0] == '-');
180     if (negative)    t = text+1;
181     else             t = text;
182
183     for (;  *t !=  '\0' ; t++)
184         {
185
186         if (*t == '.')
187         if (fract)  return INVALID_FLT;
188             else        fract = YES;
189
190         else if (isdigit(*t))
191             {
192             value = value*10 + (int)*t - (int)'0';
193             if (fract) div *= 10;
194             }
195
196         else
197             return INVALID_FLT;
198         }
199
200     if (fract)
201         value /= div;
202
203     if (negative)
204         value = -value;
205
206     return value;
207     }
208
209
210
211
212 fmt_text(s, text, width)
213     char *s, *text;
214     int width;
215     {
216
217     /* if an empty string, then all underscores */
218     if (*text == '\0')
219         for (; width > 0; width--)
220             *s++  = '_';
221
222     /* otherwise, blank fill it */
223     else
224         {
225
226         /* copy the text into the new buffer */
227         for ( ; *text != '\0';  width--)
228             *s++ = *text++;
229
230         /* fill in the rest with blanks */
231         for (;  width > 0;  width--)
232             *s++ = ' ';
233         }
234
235     /* and finally, terminate the string */
236     *s = '\0';
237     }
238
239
240
241 cvt_text(s, text)
242     char *s;
243     char *text;
244     {
245     char *lastnb;
246
247     /* skip leading blanks and underscores */
248     for (; *s == ' ' || *s == '_'; s++)
249         ;
250
251     /* copy the characters, keeping track of last
    blank or underscore */
252     lastnb = text-1;
253     for (; *s != '\0'; *text++ = *s++)
254         if (*s != ' ' && *s != '_')
255             lastnb = text;
256
257     /* truncate the text string to last nonblank
    character */
258     *(lastnb+1) = '\0';
259     }
260
261
262
263 fmtint(field, value, size, fill)
264
    /*****************************************************
    **********
265 fmtint formats an integer value into a character
    field to make the integer
266 right-justified within the character field, padded
    with leading fill
267 characters (e.g. leading blanks if a blank is passed
    in for the fill argument
268
    *****************************************************
    ********************/
269     int value;
270     char *field;
271     int size;
272     char fill;
273     {
274     int negative;
275     int dividend;
276     int remainder;
277     char *p;
278
279     /* create characters from right to left */
280     p = field + size - 1;
```

```
281
282      /* make note if this is a negative number */
283      negative = value < 0;
284      if (negative)
285          value = -value;
286
287      /* Case: Null field.  Can't do anything */
288      if (p < field)
289          ;
290
291      /* Case: value is zero.  Print a leading '0' */
292      else if (value == 0)
293          *p-- = '0';
294
295      /* Otherwise, convert each digit in turn */
296      else do
297          {
298
299          dividend = value / 10;
300          remainder = value - dividend * 10;
301          value = dividend;
302
303          *p-- = (char) (  (int)'0'  +  remainder  );
304
305          } while (p >= field && value > 0);
306
307      /* insert a minus sign if appropriate */
308      if (negative && p >= field)
309          *p-- = '-';
310
311      /* fill in leading characters */
312      while (p >= field)
313          *p-- = fill;
314      }
315
316
317  int cvtint(str)
318
 /*******************************************************
 *********
319  getint extracts an integer value from the given char-
 acter field
320  (ex: turns the string "123" into the integer 123)
321
 *******************************************************
 ********/
322      char *str;
323      {
324      int value;
325      char c;
326      int negative;
327      debug("cvtint: str=%s\n", str);
328
329      negative = (*str == '-');
330      if (negative)  str++;
331
332      /* convert the integer */
333      for (value = 0; isdigit(*str); str++)
334          value = value*10 + (int)(*str) - (int)'0';
335
336      /* if any non-digit characters, error */
337      if (*str != '\0')
338      return INVALID_NUM;
339
340      /* make negative if there was a minus sign */
341      if (negative)
342          value = -value;
343
344      debug("cvtint: value=%d\n", value);
345      return value;
346      }
347
348
349
350  fmt_phone(str, phone)
351      char str[20];
352      char *phone;
353
354      {
355      /* copy phone number and insert dashes  999999-
 999-999-9999 */
356      str[0] = phone[0];  str[1] = phone[1]; str[2] =
 phone[2];
357      str[3] = phone[3];  str[4] = phone[4]; str[5] =
 phone[5];
358      str[6] = '-';
359      str[7] = phone[6];  str[8] = phone[7]; str[9] =
 phone[8];
360      str[10] = '-';
361      str[11] = phone[9]; str[12] = phone[10]; str[13]
 = phone[11];
362      str[14] = '-';
363     str[15] = phone[12]; str[16] = phone[13]; str[17]
 = phone[14];
364      str[18] = phone[15];
365      str[19] = '\0';
366      }
367
368
369  fmt_zip(str,zip)
370      char str[20];
371      char *zip;
372      {
373      /* copy zip code and insert dashes 99999-9999 */
374      str[0] = zip[0];  str[1] = zip[1];  str[2] =
 zip[2];
375      str[3] = zip[3];  str[4] = zip[4];
376      str[5] = '-';
377      str[6] = zip[5]; str[7] = zip[6]; str[8] =
 zip[7];  str[9] = zip[8];
378      str[10] = '\0';
379      }
```

lib/iobuf.c

```
 1
 /*******************************************************
 **********************
  2  @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $
  3
  4  (c) Copyright 1996, Hewlett-Packard Company, all
 rights reserved.
  5
 *******************************************************
 *********************/
  6  #define DECLARE_IO_BUFFERS
  7  #include "iobuf.h"
  8  #undef DECLARE_IO_BUFFERS
  9  #include "tpcc.h"
 10  #include <sys/errno.h>
 11  extern int errno;
 12
 13
 14
 15  string(str)
 16      char str[];
 17      {
 18      for (; *str != '\0'; str++)
 19          pushc(*str);
 20      }
 21
 22
 23  push(str, len)
 24      char *str;
 25      int len;
 26      {
 27      for (; len > 0; len --)
 28          pushc(*str++);
 29      }
 30
 31
 32  display(scr)
 33      iobuf *scr;
 34      {
```

```
   35 /* Note: if problems doing output, let the input rou-
   tine detect it */
   36      char *p;
   37      int len;
   38      for (p = scr->beg; p < scr->end; p+=len)
   39          {
   40          len = write(1, p, scr->end - p);
   41          if (len <= 0) break;
   42          }
   43      }
   44
   45
   46  input(scr)
   47      iobuf *scr;
   48      {
   49      int len;
   50
   51      /* read in as many characters as are available */
   52      len = read(0, scr->end, scr->max - scr->end);
   53
   54      /* if end of input, then pretend we read an END
   character */
   55      if (len == 0 || (len == -1 && errno == ECONNRE-
   SET))
   56          {
   57          *scr->end = EOF;
   58          len = 1;
   59          }
   60
   61      /* Check for errors */
   62      else if (len == -1)
   63          syserror("input(scr): unable to read
   stdin\n");
   64
   65      /* update the pointers to reflect the new data */
   66      scr->end += len;
   67      *scr->end='\0';  /* for debugging */
   68      }
   69
   70
   71
   72  getkey()
   73      {
   74      if (in_buf->cur == in_buf->end)
   75          {
   76          flush();
   77          reset(in_buf);
   78          input(in_buf);
   79          }
   80
   81      return popc();
   82      }
   83
   84
```

lib/iobuf.h

```
    1
    /*******************************************************
    **********************
     2   @(#) Version: A.10.10 $Date: 96/08/06 19:33:00 $
     3
     4   (c) Copyright 1996, Hewlett-Packard Company, all
    rights reserved.
     5
    *******************************************************
    **********************/
     6
     7
    /*******************************************************
    **************
     8  History
     9  941220 LAN Added definition and initialization of the
    line_col[] array.
    10         This was needed for modifications made of cli-
    ent program to do
    11         block I/O using a WYSE terminal.
    12
    *******************************************************
    *************/
    13
    14  /* structure for screen emulation */
    15  typedef struct
    16      {
    17      int row;
    18      int col;
    19      char buf[25][81];
    20      } screen_t;
    21
    22
    23  typedef struct {
    24      char *beg;
    25      char *end;   /* for output buffers */
    26      char *max;
    27      char *cur;   /* for input buffers */
    28      } iobuf;
    29
    30  /* Macro do define an I/O buffer of x characters,
    initialized to empty */
    31  #define define_iobuf(name, size)                    \
    32  char name/**/_data[size];
    \
    33  iobuf name[1] = {{name/**/_data,
    name/**/_data,       \
    34                   name/**/_data+size, name/**/_data}}
    35
    36  #define reset(buf)  if (1) { \
    37      (buf)->cur = (buf)->end = (buf)-
    >beg;                \
    38      *(buf)->beg =
    '\0';                                    \
    39      } else (void)0
    40
    41  #define flush()  if(1) { \
    42          display(out_buf);    \
    43          reset(out_buf);     \
    44       } else (void)0
    45
    46
    47  /* Standard I/O to and from in_buf and out_buf */
    48  #ifdef DECLARE_IO_BUFFERS
    49  define_iobuf(output_stuff, 4*1024);
    50  define_iobuf(input_stuff, 1024);
    51  iobuf *in_buf = input_stuff;
    52  iobuf *out_buf = output_stuff;
    53  #else
    54  iobuf *in_buf;
    55  iobuf *out_buf;
    56  #endif
    57
    58  #define pushc(c) if (1) { \
    59      if (out_buf->end >= out_buf->max)   \
    60           error("out_buf overflow:  beg=0x%x end=%d
    max=%d\n",  \
    61            out_buf->beg, out_buf->end-out_buf-
    >beg,out_buf->max-out_buf->beg); \
    62      *(out_buf->end++) = (c);  \
    63      *(out_buf->end) = '\0';  /* debug */ \
    64      } else (void)0
    65
    66  #define popc()  \
    67      (*in_buf->cur++)
    68
    69
    70  /* Standard characters used for screen control */
    71  #define ENTER '\015'
    72  #define TAB   '\t'
    73  #define BACKTAB '\02'  /*  ^B  */
    74  #define CNTRLC '\03'
    75  #define BACKSPACE '\010'
    76  #define BELL '\07'
    77  #define BLANK ' '
    78  #define UNDERLINE '_'
    79  #define ESCAPE '\033'
```

```
 80  /*#define EOF    ((char)-1) */
 81  #define TRIGGER '\021'  /* dc1 */
 82
 83
```

lib/Makefile

```
  1
  #*******************************************************
  **********************
   2  #@(#) Version: A.10.10 $Date: 97/02/14 12:31:40 $
   3  #
   4  #(c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
   5
  #*******************************************************
  **********************
   6
   7  debug = +O2 +Ofastaccess +Oentrysched
   8  #debug = -g
   9
  10  CFLAGS= ${debug}  -Wl,-a,archive -I.
  11
  12  utils=iobuf.o delay.o errlog.o fmt.o random.o tas.o
  null_key.o null_select.o results_file.o  date.o
  prepare_socket.o shm.o spinlock.o
  13
  14  all: tpc_lib.a  server_default.o
  15
  16  tpc_lib.a: ${utils}
  17      rm -f tpc_lib.a
  18      ar -r tpc_lib.a ${utils}
  19
  20  clean:
  21      rm -f *.o
  22      rm -f *.a
  23
  24  clobber: clean
  25
  26  .s.o:
  27      cc -c $*.s
```

lib/master.h

```
  1
  /*******************************************************
  **********************
   2   @(#) Version: A.10.10 $Date: 97/03/04 15:49:41 $
   3
   4   (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
   5
  *******************************************************
  **********************/
   6
   7  #ifndef _MASTER_INCLUDED
   8  #define _MASTER_INCLUDED
   9
  10  #include "shm.h"
  11
  12  #define SHMKEY_MASTER 99999
  13  #define MAX_DRIVERS 10
  14  #define CMD "slave"
  15
  16
  17
  /*******************************************************
  ***********
  18
  19
  *******************************************************
  ***********/
  20
  21  shm_t *shm_master;    /* master shared memory pointer
  */
  22  int sd[MAX_DRIVERS];  /* socket descriptors */
  23
  24  #define m_state shm_master->s_state
  25  #define m_desired shm_master->s_desired
  26  #define m_attached shm_master->s_attached
  27  #define m_transactions shm_master->s_transactions
  28  #define m_max_transactions shm_master-
  >s_max_transactions
  29  #define m_max_duration shm_master->s_max_duration
  30
  31  typedef struct
  32  {
  33      struct
  34      {
  35          int t_desired;
  36          int t_state;
  37          struct timeval t_start_time;
  38      } out_data;
  39
  40      struct
  41      {
  42          int t_attached;
  43      int t_transactions;
  44      int t_max_transactions;
  45      int t_max_duration;
  46          stat_t t_stat[1];
  47      } in_data;
  48
  49  } xfer_data_t;
  50
  51  xfer_data_t xfer_data;
  52
  53
  54  typedef struct
  55  {
  56      int     s_attached;
  57      int     s_transactions;
  58      int     s_max_transactions;
  59      int     s_max_duration;
  60      stat_t  s_stat[1];
  61  } store_data_t;
  62
  63  store_data_t str_data;
  64
  65  #define d_desired xfer_data.out_data.t_desired
  66  #define d_state xfer_data.out_data.t_state
  67  #define d_start_time xfer_data.out_data.t_start_time
  68  #define d_attached xfer_data.in_data.t_attached
  69  #define d_max_transactions
  xfer_data.in_data.t_max_transactions
  70  #define d_max_duration
  xfer_data.in_data.t_max_duration
  71  #define d_transactions
  xfer_data.in_data.t_transactions
  72  #define d_stat xfer_data.in_data.t_stat[0]
  73
  74
  75  #endif /* _MASTER_INCLUDED */
```

lib/null_key.c

```
  1
  /*******************************************************
  **********************
   2   @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $
   3
   4   (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
   5
  *******************************************************
  **********************/
   6  /* dummy keystroke routines */
   7  neworder_key(){}
   8  payment_key(){}
   9  ordstat_key(){}
  10  delivery_key(){}
  11  stocklev_key(){}
```

lib/null_select.c

```
   1
   /*****************************************************
   **********************
    2  @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $
    3
    4  (c) Copyright 1996, Hewlett-Packard Company, all
   rights reserved.
    5
   *****************************************************
   **********************/
    6 /* dummy menu selection routines */
    7 neworder_select(){}
    8 payment_select(){}
    9 ordstat_select(){}
   10 delivery_select(){}
   11 stocklev_select(){}
```

lib/odbc.h

```
   1  /* ODBC.H */
    2
    3 #include <sql.h>
    4 #include <sqlext.h>
    5
    6 #define DEL_FIFO "/dev/delivery"
    7
    8 #ifndef TRUE
    9 #define TRUE 1
   10 #endif
   11
   12 #ifndef FALSE
   13 #define FALSE 0
   14 #endif
   15
   16 #define DEADLOCK -2
   17
   18 #ifndef MaxTries
   19 #define MaxTries 25
   20 #endif
   21
   22 #define LinesPerCall 15
   23
   24 #define dsn (unsigned char *) "MSSqlServer"
   25 #define user (unsigned char *) "sa"
   26 #define server (unsigned char *) "HPDPC338"
   27 #define database (unsigned char *) "tpcc"
   28 #define passwd (unsigned char *) ""
   29
   30 HDBC hdbc;
   31 HENV henv;
   32 HSTMT hstmt;
   33
   34 /* For error checking */
   35 extern char errorname[32];
   36
   37 int prev_xact_type;
   38 short o_ol_cnt, o_ol_done, o_ol_now;
   39 short o_all_local;
   40 short lines_per_call;
   41
   42 #define XACT_NEWO 0
   43 #define XACT_PAYM 1
   44 #define XACT_ORDS 2
   45 #define XACT_DEL 3
   46 #define XACT_STOCK 4
   47 #define XACT_BKEND 5
   48
   49 /*
   50 ** Define the TPC-C functions
   51 */
   52
   53 int ODBCError();
   54
   55 int new_order_rpc();
   56 int payment_rpc();
   57 int order_status_rpc();
   58 int stock_level_rpc();
   59 int delivery_rpc();
   60 int connect_odbc_user();
   61
   62 void display_xction();
   63 void sleep_before_retry ();
   64 void pick_xact_type();
```

lib/prepare_socket.c

```
   1
   /*****************************************************
   **********************
    2   @(#) Version: A.10.10 $Date: 96/04/02 16:26:26 $
    3
    4   (c) Copyright 1996, Hewlett-Packard Company, all
   rights reserved.
    5
   *****************************************************
   **********************/
    6  #include <sys/socket.h>
    7  #include <netinet/in.h>
    8  #include <netinet/tcp.h>
    9  #include <netdb.h>
   10
   11  int prepare_socket(fd)
   12      int fd;
   13      {
   14      int yes = 1;
   15
   16      if (setsockopt(fd, SOL_SOCKET, SO_KEEPALIVE,
   &yes, sizeof(yes)) < 0)
   17          return -1;
   18      if (setsockopt(fd, SOL_SOCKET, SO_REUSEADDR,
   &yes, sizeof(yes)) < 0)
   19          return -1;
   20     if (setsockopt(fd, SOL_SOCKET, TCP_NODELAY, &yes,
   sizeof(yes)) < 0)
   21      return -1;
   22      /* SO_SNDBUF, SO_RCVBUF, TPC_MAXSEG  should be
   set ?? */
   23      return 0;
   24      }
   25
```

lib/random.c

```
   1
   /*****************************************************
   **********************
    2   @(#) Version: A.10.10 $Date: 97/02/20 11:47:10 $
    3
    4   (c) Copyright 1996, Hewlett-Packard Company, all
   rights reserved.
    5
   *****************************************************
   **********************/
    6  #include "tpcc.h"
    7  #include "string.h"
    8  #include "random.h"
    9
   10  double drand48();
   11
   12  char lastNames[1000][16];
   13  char customerData1[10][301];
   14  char customerData2[10][201];
   15  char stockData1[10][27];
   16  char stockData2[10][25];
   17  char historyData1[10][13];
   18  char historyData2[10][13];
   19  char citystreetData1[10][11];
   20  char citystreetData2[10][11];
   21  char firstNameData1[10][9];
   22  char firstNameData2[10][9];
```

```
23  char StockDistrict[10][25];
24  char phoneData[10][17];
25
26  void GenerateLastNames()
27  {
28      int      i;
29      char *name;
30      static char *n[] = {"BAR", "OUGHT", "ABLE",
"PRI", "PRES",
31                          "ESE", "ANTI", "CALLY",
"ATION", "EING"};
32
33      for(i = 0; i < 1000; i++) {
34          name = &lastNames[i];
35          strcpy(name, n[(i/100)%10]);
36          strcat(name, n[(i/10) %10]);
37          strcat(name, n[(i/1)  %10]);
38      }
39  }
40
41  int MakeNumberString(min, max, num)
42      int min;
43      int max;
44      TEXT num[];
45      {
46      static char digit[]="0123456789";
47      int length;
48      int i;
49
50      length = RandomNumber(min, max);
51
52      for (i=0;  i<length;  i++)
53          num[i] = digit[RandomNumber(0,9)];
54      num[length] = '\0';
55
56      return length;
57      }
58
59  ID RandomWarehouse(local, scale, percent)
60      ID local;
61      ID scale;
62      int percent;   /* percent of remote transactions
*/
63      {
64      ID w_id;
65
66      /* For the given percent of the time, pick the
local warehouse */
67      if (RandomNumber(1, 100) > percent || scale == 1)
68          w_id = local;
69
70      /* Otherwise, pick a non-local warehouse */
71      else
72          {
73          w_id = RandomNumber(2, scale);
74          if (w_id == local)
75              w_id = 1;
76          }
77      return w_id;
78      }
79
80
81  /* Initialize a table of Random strings for the
stock-district
82     field in the stock table.  We can use a table of 10
elements
83     and select randomly from this table via rule
4.3.2.2 in
84     the TPC-C spec */
85  void InitRandomStrings()
86  {
87      int i;
88
89      for (i=0; i < 10; i++) {
90          MakeAlphaString(24,24,&StockDistrict[i]);
91
92          MakeAlphaString(300,300,&customerData1[i]);
93          MakeAlphaString(0,200,&customerData2[i]);
94
95          MakeAlphaString(26,26,&stockData1[i]);
96          MakeAlphaString(0,24,&stockData2[i]);
97
98          MakeAlphaString(12,12,&historyData1[i]);
99          MakeAlphaString(0,12, &historyData2[i]);
100
101         MakeAlphaString(10,10,&citystreetData1[i]);
102         MakeAlphaString(0 ,10,&citystreetData2[i]);
103
104         MakeAlphaString(8,8,&firstNameData1[i]);
105         MakeAlphaString(0,8,&firstNameData2[i]);
106
107         MakeNumberString(16,16,&phoneData[i]);
108     }
109     GenerateLastNames();
110 }
111
112 int MakeAlphaString(min, max, str)
113     int min;
114     int max;
115     TEXT str[];
116     {
117     static char character[] = "abcdefghijklmnopqrstu-
 vwxyz";
118     int length;
119     int i;
120
121     length = RandomNumber(min, max);
122
123     for (i=0;  i<length;  i++) {
124         /* NOTE:  we use sizeof(character)-2 because
 of the following:
125            subtract 1 because we are numbering from 0
 instead of 1 and
126            subtract 1 because the sizeof(character) is
 1 greater than
127             the data in character because of the ini-
 visible C string
128             terminator at the end. */
129         str[i] = character[RandomNumber(0,
 sizeof(character)-2)];
130     }
131     str[length] = '\0';
132
133     return length;
134     }
135
136 void RandomPermutation(perm, n)
137     int perm[];
138     int n;
139     {
140     int i, r, t;
141
142     /* generate the identity permutation to start
 with */
143     for (i=1; i<=n; i++)
144         perm[i] = i;
145
146     /* randomly shuffle the permutation */
147     for (i=1; i<=n; i++)
148         {
149         r = RandomNumber(i, n);
150         t = perm[i]; perm[i] = perm[r]; perm[r] = t;
151         }
152     }
153
154
155 void RandomDelay(mean, adjust)
156
 /*****************************************************
 ************
157 random_sleep sleeps according to the TPC specifica-
 tion
158
 *****************************************************
```

```
 ***********/
159      double mean;
160      double adjust;
161      {
162      double secs;
163      double exponential();
164
165      secs = exponential(mean);
166
167      delay(secs+adjust);
168      }
169
170  double exponential(mean)
171
 /*******************************************************
 ***********
172  exponential generates a reverse exponential distribu-
 tion
173
 *******************************************************
 **************/
174      double mean;
175      {
176      double x;
177      double log();
178
179      x = -log(1.0-drand48()) * mean;
180
181      return x;
182      }
183
184
185  void Randomize()
186      {
187      srand48(time(0)+getpid());
188      }
```

lib/random.h

```
  1
 /*******************************************************
 **********************
  2   @(#) Version: A.10.10 $Date: 97/02/20 11:47:20 $
  3
  4   (c) Copyright 1996, Hewlett-Packard Company, all
 rights reserved.
  5
 *******************************************************
 **********************/
  6  #ifndef TPCC_RANDOM
  7  #define TPCC_RANDOM
  8
  9  double drand48();
 10
 11  extern int    MakeNumberString();
 12  extern ID     RandomWarehouse();
 13  extern int    MakeAlphaString();
 14  extern void   RandomPermutation();
 15  extern void   RandomDelay();
 16  extern double exponential();
 17  extern void   Randomize();
 18
 19  extern char lastNames[1000][16];
 20  extern char customerData1[10][301];
 21  extern char customerData2[10][201];
 22  extern char stockData1[10][27];
 23  extern char stockData2[10][25];
 24  extern char historyData1[10][13];
 25  extern char historyData2[10][13];
 26  extern char citystreetData1[10][11];
 27  extern char citystreetData2[10][11];
 28  extern char firstNameData1[10][9];
 29  extern char firstNameData2[10][9];
 30  extern char StockDistrict[10][25];
 31  extern char phoneData[10][17];
 32
 33
 /*******************************************************
 ******************
34  RandomNumber selects a uniform random number from min
to max inclusive
35
 *******************************************************
 *****************/
36  #define RandomNumber(min,max)  \
37       ((int)(drand48() * ((int)(max) - (int)(min) + 1))
+ (int)(min))
38
39
 /*******************************************************
 ***************
40  NURandomNumber selects a non-uniform random number
41
 *******************************************************
 **************/
42  #define NURandomNumber(a, min, max, c) \
43       ((RandomNumber(0, a) | RandomNumber(min, max)) +
(c)) % \
44            ((max) - (min) + 1) + (min)
45
46  #define SelectHistoryData(data) \
47  { \
48       strcpy(data,historyData1[RandomNumber(0,9)]);\
49       strcat(data,historyData2[RandomNumber(0,9)]);\
50  }
51
52  #define SelectCityStreetData(data) \
53  { \
54       strcpy(data,citystreetData1[RandomNumber(0,9)]);
\
55       strcat(data,citystreetData2[RandomNumber(0,9)]);
\
56  }
57
58  #define SelectFirstName(data) \
59  { \
60       strcpy(data,firstNameData1[RandomNumber(0,9)]); \
61       strcat(data,firstNameData2[RandomNumber(0,9)]); \
62  }
63
64  #define SelectHistoryData(data) \
65  { \
66       strcpy(data,historyData1[RandomNumber(0,9)]); \
67       strcat(data,historyData2[RandomNumber(0,9)]); \
68  }
69
70  #define SelectStockData(data) \
71  { \
72       strcpy(data,stockData1[RandomNumber(0,9)]); \
73       strcat(data,stockData2[RandomNumber(0,9)]); \
74  }
75
76  #define SelectClientData(data) \
77  { \
78       strcpy(data,customerData1[RandomNumber(0,9)]); \
79       strcat(data,customerData2[RandomNumber(0,9)]); \
80  }
81
82  #define SelectPhoneData(data) strcpy(data,phone-
Data[RandomNumber(0,9)])
83  #define SelectStockDistrict(data) strcpy(data,Stock-
District[RandomNumber(0,9)])
84
85  #define MakeZip(zip) \
86  { \
87     MakeNumberString(4, 4, zip); \
88     zip[4] = '1'; \
89     zip[5] = '1'; \
90     zip[6] = '1'; \
91     zip[7] = '1'; \
92     zip[8] = '1'; \
93     zip[9] = '\0'; \
94  }
95
```

```
 96  #define MakeAddress(str1, str2, city, state, zip) \
 97  { \
 98      SelectCityStreetData(str1); \
 99      SelectCityStreetData(str2); \
100      SelectCityStreetData(city); \
101      MakeAlphaString(2,2,state); \
102      MakeZip(zip); \
103  }
104
105  #define LastName(num, name) strcpy(name, last-
  Names[(num)])
106
107  #define Original(str) \
108  { \
109      int len = strlen(str); \
110      if (len >= 8) { \
111          int pos = RandomNumber(0,(len-8)); \
112          str[pos+0] = 'O'; \
113          str[pos+1] = 'R'; \
114          str[pos+2] = 'I'; \
115          str[pos+3] = 'G'; \
116          str[pos+4] = 'I'; \
117          str[pos+5] = 'N'; \
118          str[pos+6] = 'A'; \
119          str[pos+7] = 'L'; \
120      } \
121  }
122
123
124  #endif
125
```

lib/results_file.c

```
  1
  /*********************************************************
  **********************
   2  @(#) Version: A.10.10 $Date: 96/08/06 11:56:24 $
   3
   4  (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
   5
  *********************************************************
  **********************/
   6 #include <unistd.h>
   7 #include <stdio.h>
   8 #include "tpcc.h"
   9
  10
  11
  12
  13 static FILE *rfile;
  14
  15 results_open(id)
  16      int id;
  17      {
  18      char fullname[128];
  19      char *basename;
  20
  21     /* get the base file name for the deferred results
  */
  22      /*
  23       * Make it a directory under /tmp so at least we
  can set it to a
  24       * symbolic link in case /tmp doesn't have enough
  room.
  25       */
  26      basename = getenv("TPCC_RESULTS_FILE");
  27      if (basename == NULL)
  28          basename = "/tmp/TPCC_RESULTS_FILE";
  29
  30      /* create the full file name */
  31      sprintf(fullname, "%s.%d", basename, id);
  32
  33      /* open the file */
  34      unlink(fullname);
  35      rfile = fopen(fullname, "wb");
  36      if (rfile == NULL)
  37          syserror("Delivery server %d can't open file
  %s\n", id, fullname);
  38
  39      /* allocate a larger buffer */
  40      }
  41
  42
  43
  44 results(t)
  45      delivery_trans *t;
  46      {
  47      if (fwrite(t, sizeof(*t), 1, rfile) != 1)
  48          syserror("Delivery server: Can't post
  results\n");
  49      }
  50
  51
  52 results_close()
  53      {
  54      if (fclose(rfile) < 0)
  55          syserror("Delivery server can't close
  file\n");
  56      }
  57
```

lib/server_default.c

```
1
/*********************************************************
**********************
    2 @(#) Version: A.10.10 $Date: 96/04/02 16:26:26 $
    3
    4 (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
    5
*********************************************************
**********************/
    6int server_default = 1;
```

lib/shm.c

```
   1  #include <stdio.h>
   2  #include <values.h>
   3  #include <unistd.h>
   4  #include <sys/types.h>
   5  #include <sys/shm.h>
   6  #include "shm.h"
   7
   8  void *
   9  attach_shm(key, size, create)
  10
  /*********************************************************
  *************
  11  attach_shm attaches to the shared memory
  12
  *********************************************************
  ************/
  13  key_t   key;
  14  size_t size;
  15  int    create;
  16      {
  17      void *ptr;
  18      int shmid;
  19      int flag;
  20
  21      /* create the shm if it doesn't already exist */
  22      flag = (create) ? 0777|IPC_CREAT : 0777;
  23
  24      shmid = shmget(key, size, flag);
  25      if (shmid < 0) {
  26      if (create) {
  27             syserror("Can't create shared memory.\n");
  28      } else {
  29              return NULL;
  30      }
```

```
    }

    /* attach to the shared memory */
    ptr = shmat(shmid, NULL, 0);
    if (ptr == (void *)-1)
        syserror("Can't attach shared memory\n");

    return ptr;
}

void
detach_shm(ptr)

/*********************************************************
*******
cleanup_shm detaches from the shared memory region

*********************************************************
******/
void *ptr;
    {
    shmdt((void *)ptr);
    }
```

lib/shm.h

```
/*********************************************************
**********************
 @(#) Version: A.10.10 $Date: 97/02/14 12:33:11 $

 (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.

*********************************************************
**********************/
#ifndef SHM_INCLUDED
#define SHM_INCLUDED

#include <time.h>
#include "tpcc.h"
#include "shm_lookup.h"

#define SHMKEY_DRIVER 55555
shm_t *shm;  /* pointer to shared memory for each
driver */

int ndrivers;
int drivernum;
int nclients;
int clientnum;

extern void *attach_shm();
extern void detach_shm();

#endif /* SHM_INCLUDED */
```

lib/shm_lookup.h

```
/*********************************************************
**********************
 @(#) Version: A.10.10 $Date: 97/02/14 12:33:40 $

 (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.

*********************************************************
**********************/
#ifndef SHM_LOOKUP_INCLUDED
#define SHM_LOOKUP_INCLUDED

#include <time.h>
#include "tpcc.h"
#include "spinlock.h"

typedef struct
    {
    char *old;
    char *next;
    char buffer[4000000];   /* big enough for two+
seconds of data */
    } success_buf;;

#define S_FIRST  (shm->success.buffer)
#define S_LAST   (shm->success.buffer + sizeof(shm-
>success.buffer))
#define S_NEXT   shm->success.next      /* points to
next slot to fill */
#define S_OLD    shm->success.old       /* points to
last unfilled slot */

/* states to sync with multiple drivers */
/*
 * WAIT -- driver is waiting for all users to login
 * RUN  -- all users logged in, running transactions
 * STOP -- test is over, stop
 * CONFIGURE -- driver is in process of configuring
shared memory
 */

#define WAIT      0
#define RUN       1
#define STOP      2
#define CONFIGURE 3

typedef struct {
    double total_response[6];
    int count[6];
    int bad[6];
    } stat_t;


/* Now, the entire shared memory declaration */
typedef struct
    {

    /* control information */
    lock_t           s_shm_lock; /* spin lock for
accessing shared memory */
    volatile int     s_state;    /* state of the
machine */
    volatile int     s_stopflag;
    volatile int     s_desired;  /* # users we want
attached */
    volatile int     s_attached; /* # users we know
are attached */
    volatile int    s_spawned;  /* # processes active
that could attach */
    volatile int    s_transactions; /* # transactions
so far */
/*  volatile int   s_sync; *//* synchronize the log
output after every trans */
    volatile int     s_first_user;

    /* configuration parameters */
    volatile int     s_scale;
    volatile int     s_max_transactions;
    volatile int     s_max_duration;
    volatile double  s_fudge;
    volatile int    s_server;        /* tpca or tpcb
*/
    volatile int     s_dbg;           /* single user
debugging */
    volatile double  s_think_factor;
    volatile int     s_post_transactions;

    /* run time statistics */
    stat_t s_stat[1];
    struct timeval start_time;

    /* success buffer */
    success_buf success;
```

```
 76
 77      } shm_t;
 78
 79  #define first_user shm->s_first_user
 80  #define shm_lock shm->s_shm_lock
 81  #define state shm->s_state
 82  #define file_lock shm->s_log.l_file_lock
 83  #define attached shm->s_attached
 84  #define spawned shm->s_spawned
 85  #define desired shm->s_desired
 86  #define stopflag shm->s_stopflag
 87  #define scale shm->s_scale
 88  #define max_transactions shm->s_max_transactions
 89  #define max_duration shm->s_max_duration
 90  #define fudge  shm->s_fudge
 91  #define transactions shm->s_transactions
 92  #define sync shm->s_sync
 93  #define server shm->s_server
 94  #define stat   shm->s_stat
 95  #define dbg    shm->s_dbg
 96  #define think_factor shm->s_think_factor
 97  #define post_transactions shm->s_post_transactions
 98
 99  extern shm_t *shm;
100  extern int shmid;
101
102  extern int ndrivers;
103  extern int drivernum;
104  extern int nclients;
105  extern int clientnum;
106
107  #endif /* SHM_INCLUDED */
```

lib/spinlock.c

```
  1
  /*********************************************************
  **********************
  2   @(#) Version: A.10.10 $Date: 97/02/14 12:33:58 $
  3
  4   (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
  5
  *********************************************************
  **********************/
  6  #include "spinlock.h"
  7
  8
  9
  /*********************************************************
  **********
 10
  *********************************************************
  *********
 11
 12  Locking routines to give exclusive access to shared
  memory
 13
 14
  *********************************************************
  **********
 15
  *********************************************************
  ***********/
 16
 17  void
 18  lock(latch)
 19
  /*********************************************************
  *******
 20  lock gets exclusive access to the shared memory data
  structures
 21
  *********************************************************
  *******/
 22      lock_t latch;
 23      {
 24      while (!tas(addr64(latch)))
 25          delay(.01);
 26      }
 27
 28
 29  void
 30  unlock(latch)
 31
  /*********************************************************
  ****
 32  unlock releases exclusive access to shmem
 33
  *********************************************************
  ****/
 34      lock_t latch;
 35      {
 36      *addr64(latch) = 1;
 37      }
 38
 39
 40  void
 41  init_lock(latch)
 42
  /*********************************************************
  ********
 43  init_lock initializes the lock structure
 44
  *********************************************************
  ********/
 45      lock_t latch;
 46      {
 47      unlock(latch);
 48      }
 49
```

lib/spinlock.h

```
  1
  /*********************************************************
  **********************
  2   @(#) Version: A.10.10 $Date: 97/02/14 12:34:10 $
  3
  4   (c) Copyright 1996, Hewlett-Packard Company, all
  rights reserved.
  5
  *********************************************************
  **********************/
  6  #ifndef SPINLOCK_HEADER_INCLUDED
  7  #define SPINLOCK_HEADER_INCLUDED
  8
  9  typedef struct
 10      {
 11      int wx[32];
 12      int count;
 13      int conflict;
 14      int sleep;
 15      } lock_t[1];
 16
 17  /* macro to point to the relevant lock word */
 18  #define addr64(latch)     ((int *)(((int)latch + 63)
  & ~63))
 19
 20  extern void lock();
 21  extern void unlock();
 22  extern void init_lock();
 23
 24  #endif
```

lib/tas.s

```
  1
  2
  ;*********************************************************
  **********************
  3  ; @(#) Version: A.10.10 $Date: 96/04/02 16:27:37 $
  4  ;
```

```
 5  ; (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 6
;*******************************************************
**********************
 7      .code
 8
;*******************************************************
******************
 9  ; Test and set routines implemented with the LDCWS
instruction
10  ;
11  ; The LDCWS instruction is very similar to a tradi-
tional test+set instruction
12  ; with the following exceptions:
13  ;    o Zero means set and One means clear.
14  ;    o Word must be 16 byte aligned
15  ;
16  ; Acheiving 16 byte alignment is awkward in C, so
these routines have
17  ; been designed to take a larger unaligned structure
and round up to the
18  ; first 16 byte aligned word of the structure.
19  ;
20  ; A reasonable C declaration for this structure could
be:
21  ;      typedef  struct { int words[4]; }  latch_t.
22  ;
23  ; On future multiprocessor  machines,  normal load and
store instructions could
24  ; be reordered arbitrarily by the hardware.  The
stws,ma 0() and ldws,ma 0()
25  ; instructions will force a synchronization of reor-
dered loads and stores.
26
;*******************************************************
****************
27
28
29
30  ; tas(latch)
31
;*****************************************************
32  ; tas is true if we succeeded in acquiring the latch
33  ;*****************************************************
34      .proc
35      .callinfo
36      .export tas
37  tas
38
39  ; test+set, and return
40      bv      (rp)
41      ldcws   (arg0), ret0
42      .procend
```

lib/tpcc.h

```
 1
/*******************************************************
**********************
 2   @(#) Version: A.10.10 $Date: 96/07/11 16:52:21 $
 3
 4   (c) Copyright 1996, Hewlett-Packard Company, all
rights reserved.
 5
*******************************************************
**********************/
 6  #ifndef TPCC_INCLUDED
 7  #define TPCC_INCLUDED
 8  #include <values.h>
 9
10
11  /* The auditor can define these 20 char strings to be
anything */
12  #define DRIVER_AUDIT_STRING  "driver audit string"
13  #define CLIENT_AUDIT_STRING  "client audit string"
14
15  #ifdef DEBUG
16  #define debug printf
17  #else
18  #define debug (void)
19  #endif
20
21  #include <stdio.h>
22
23  typedef int ID;            /* All id's */
24  typedef double MONEY;      /* Large integer number of
cents */
25  typedef char TEXT;         /* Add an extra byte for
null terminator */
26  typedef double TIME;       /* Elapsed seconds from
start of run (float?) */
27  typedef int COUNT;        /* integer numbers of things
*/
28  typedef double REAL;       /* real numbers */
29  typedef int LOGICAL;       /* YES or NO */
30  typedef struct {           /* days and seconds since
Jan 1, 1900 */
31      int day;               /* NULL represented by neg-
ative day */
32      int sec;
33      } DATE;
34
35  /* Macro to convert time of day to TIME */
36  #include <time.h>
37  extern struct timeval start_time;
38  #define elapsed_time(t) ( ((t)->tv_sec -
start_time.tv_sec) +  \
39                            ((t)->tv_usec -
start_time.tv_usec) / 1000000.0 )
40
41  typedef enum {Num,Money,Text,Time,Real,Date}
FIELD_TYPE;   /* screen field types */
42
43
44  /* Various TPCC constants */
45  #define W_ID_LEN        4
46  #define D_ID_LEN        2
47  #define C_ID_LEN        4
48  #define I_ID_LEN        6
49  #define OL_QTY_LEN      2
50  #define PMT_LEN         7
51  #define C_ID_LEN        4
52  #define C_LAST_LEN      16
53  #define CARRIER_LEN     2
54  #define THRESHOLD_LEN   2
55  #define DIST_PER_WARE   10
56  #define CUST_PER_DIST   3000
57  #define ORD_PER_DIST    3000
58  #define MAXITEMS        100000
59  #define MAX_DIGITS      3       /* # of digits of the
NURand number selected
60                                     to generate the
customer last name */
61  #define MAXWAREHOUSE   2000     /* maximum # of ware-
houses - scaling factor */
62  #define LOADSEED      42 /* # of digits of the NURand
number selected
63
64
/*******************************************************
*************/
65  /* database identifiers and popula-
tions                        */
66
/*******************************************************
*************/
67
68    int no_warehouse;                        /*  scal-
ing factor    */
69    int no_item;                        /* 100000
*/
70    int no_dist_pw;                      /*    10
*/
```

```
 71   int no_cust_pd;                    /*   3000
 */
 72   int no_ord_pd;                     /*   3000
 */
 73   int no_new_pd;                     /*    900
 */
 74   int tpcc_load_seed;                /*    900
 */
 75
 76  /* fields to add to each transaction for acid testing
 */
 77  #define ACID_STUFF     \
 78      char acid_txn[2];  \
 79      int  acid_timing;  \
 80      int  acid_action;  \
 81      FILE *acid_res
 82
 83  typedef struct {
 84      ID OL_SUPPLY_W_ID;
 85      ID OL_I_ID;
 86      TEXT I_NAME[24+1];
 87      COUNT OL_QUANTITY;
 88      COUNT S_QUANTITY;
 89      MONEY I_PRICE;
 90      char brand_generic;
 91      } neworder_item;
 92
 93  typedef struct {
 94      int status;
 95      LOGICAL all_local;
 96      ID W_ID;
 97      ID D_ID;
 98      ID C_ID;
 99      TEXT C_LAST[C_LAST_LEN+1];
100      TEXT C_CREDIT[2+1];
101      REAL C_DISCOUNT;
102      COUNT O_OL_CNT;
103      ID O_ID;
104      TEXT O_ENTRY_D[20];  /* dates as text fields */
105      REAL W_TAX;
106      REAL D_TAX;
107      neworder_item item[15];
108      ACID_STUFF;
109      } neworder_trans;
110
111
112
113  typedef struct {
114      int status;
115      LOGICAL byname;
116      ID W_ID;
117      ID D_ID;
118      ID C_ID;
119      ID C_D_ID;
120      ID C_W_ID;
121      MONEY H_AMOUNT;
122      TEXT H_DATE[20];  /* date as text field */
123      TEXT W_STREET_1[20+1];
124      TEXT W_STREET_2[20+1];
125      TEXT W_CITY[20+1];
126      TEXT W_STATE[2+1];
127      TEXT W_ZIP[9+1];
128      TEXT D_STREET_1[20+1];
129      TEXT D_STREET_2[20+1];
130      TEXT D_CITY[20+1];
131      TEXT D_STATE[2+1];
132      TEXT D_ZIP[9+1];
133      TEXT C_FIRST[16+1];
134      TEXT C_MIDDLE[2+1];
135      TEXT C_LAST[16+1];
136      TEXT C_STREET_1[20+1];
137      TEXT C_STREET_2[20+1];
138      TEXT C_CITY[20+1];
139      TEXT C_STATE[2+1];
140      TEXT C_ZIP[9+1];
141      TEXT C_PHONE[16+1];
142      TEXT C_SINCE[20]; /* date as text field */
143      TEXT C_CREDIT[2+1];
144      MONEY C_CREDIT_LIM;
145      REAL C_DISCOUNT;
146      REAL C_BALANCE;
147      TEXT  C_DATA[200+1];
148      ACID_STUFF;
149      } payment_trans;
150
151
152  typedef struct {
153      int status;
154      LOGICAL byname;
155      ID W_ID;
156      ID D_ID;
157      ID C_ID;
158      TEXT C_FIRST[16+1];
159      TEXT C_MIDDLE[2+1];
160      TEXT C_LAST[16+1];
161      MONEY C_BALANCE;
162      ID O_ID;
163      TEXT O_ENTRY_DATE[20];   /* date as text field */
164      ID O_CARRIER_ID;
165      COUNT ol_cnt;
166      struct {
167          ID OL_SUPPLY_W_ID;
168          ID OL_I_ID;
169          COUNT OL_QUANTITY;
170          MONEY OL_AMOUNT;
171          TEXT OL_DELIVERY_DATE[20]; /* date as text
 field */
172          } item[15];
173      ACID_STUFF;
174      } ordstat_trans;
175
176
177  typedef struct {
178      int status;
179      ID W_ID;
180      ID D_ID;
181      COUNT threshold;
182      COUNT low_stock;
183      ACID_STUFF;
184      } stocklev_trans;
185
186  typedef struct {
187      int status;
188      ID W_ID;
189      ID O_CARRIER_ID;
190      struct {
191          ID O_ID;
192      int status;
193          } order[10];
194      struct timeval enque[1];
195      struct timeval deque[1];
196      struct timeval complete[1];
197      ACID_STUFF;
198      } delivery_trans;
199
200
201  typedef union {
202      neworder_trans neworder;
203      payment_trans payment;
204      ordstat_trans ordstat;
205      delivery_trans delivery;
206      stocklev_trans stocklev;
207      int status;
208      } generic_trans;
209
210
211
 /*******************************************************
212  Record formats for results
213
 *******************************************************
 /
214
215  #ifdef NOTYET
```

```
216  typedef struct
217      {
218      float t1, t2, t3, t4, t5;
219      int status           :8;
220      unsigned int type    :3;
221      unsigned int ol_cnt  :4;
222      unsigned int remote_ol_cnt :4;
223      unsigned int byname :1;
224      unsigned int remote  :1;
225      unsigned int skipped  :4;
226      } success_t;
227  #endif
228
229  typedef struct
230      {
231      TIME t1, t2, t3, t4, t5;
232      int status;
233      unsigned int type    :3;
234      unsigned int ol_cnt  :4;
235      unsigned int remote_ol_cnt :4;
236      unsigned int byname :1;
237      unsigned int remote  :1;
238      unsigned int skipped  :4;
239      } success_t;
240
241  typedef struct
242      {
243      struct timeval start_time;
244      } success_header_t;
245
246
247
 /*******************************************************
 *****************
248  Record formats for loading routines.  (DB's have own
 internal formats
249
 *******************************************************
 ***************/
250  typedef struct
251      {
252      ID W_ID;
253      TEXT W_NAME[10+1];
254      TEXT W_STREET_1[20+1];
255      TEXT W_STREET_2[20+1];
256      TEXT W_CITY[20+1];
257      TEXT W_STATE[2+1];
258      TEXT W_ZIP[9+1];
259      REAL W_TAX;
260      MONEY W_YTD;
261      } warehouse_row;
262
263  typedef struct
264      {
265      ID D_ID;
266      ID D_W_ID;
267      TEXT D_NAME[10+1];
268      TEXT D_STREET_1[20+1];
269      TEXT D_STREET_2[20+1];
270      TEXT D_CITY[20+1];
271      TEXT D_STATE[2+1];
272      TEXT D_ZIP[9+1];
273      REAL D_TAX;
274      MONEY D_YTD;
275      ID D_NEXT_O_ID;
276      } district_row;
277
278
279  typedef struct
280      {
281      ID C_ID;
282      ID C_D_ID;
283      ID C_W_ID;
284      TEXT C_FIRST[16+1];
285      TEXT C_MIDDLE[2+1];
286      TEXT C_LAST[16+1];
287      TEXT C_STREET_1[20+1];
288      TEXT C_STREET_2[20+1];
289      TEXT C_CITY[20+1];
290      TEXT C_STATE[2+1];
291      TEXT C_ZIP[9+1];
292      TEXT C_PHONE[16+1];
293      DATE C_SINCE[20];
294      TEXT C_CREDIT[2+1];
295      MONEY C_CREDIT_LIM;
296      REAL C_DISCOUNT;
297      MONEY C_BALANCE;
298      MONEY C_YTD_PAYMENT;
299      COUNT C_PAYMENT_CNT;
300      COUNT C_DELIVERY_CNT;
301      TEXT  C_DATA[500+1];
302      } customer_row;
303
304  typedef struct
305      {
306      ID H_C_ID;
307      ID H_C_D_ID;
308      ID H_C_W_ID;
309      ID H_D_ID;
310      ID H_W_ID;
311      DATE H_DATE[20];
312      MONEY H_AMOUNT;
313      TEXT H_DATA[24+1];
314      } history_row;
315
316  typedef struct
317      {
318      ID NO_O_ID;
319      ID NO_D_ID;
320      ID NO_W_ID;
321      } neworder_row;
322
323  typedef struct
324      {
325      ID O_ID;
326      ID O_D_ID;
327      ID O_W_ID;
328      ID O_C_ID;
329      DATE O_ENTRY_D[20];
330      ID O_CARRIER_ID;
331      COUNT O_OL_CNT;
332      LOGICAL O_ALL_LOCAL;
333      } order_row;
334
335  typedef struct
336      {
337      ID OL_O_ID;
338      ID OL_D_ID;
339      ID OL_W_ID;
340      ID OL_NUMBER;
341      ID OL_I_ID;
342      ID OL_SUPPLY_W_ID;
343      DATE OL_DELIVERY_D;
344      COUNT OL_QUANTITY;
345      MONEY OL_AMOUNT;
346      TEXT OL_DIST_INFO[24+1];
347      } orderline_row;
348
349  typedef struct
350      {
351      ID I_ID;
352      ID I_IM_ID;
353      TEXT I_NAME[24+1];
354      MONEY I_PRICE;
355      TEXT I_DATA[50+1];
356      } item_row;
357
358  typedef struct
359      {
360      ID S_I_ID;
361      ID S_W_ID;
362      COUNT S_QUANTITY;
363      TEXT S_DIST_01[24+1];
364      TEXT S_DIST_02[24+1];
```

```
365       TEXT S_DIST_03[24+1];
366       TEXT S_DIST_04[24+1];
367       TEXT S_DIST_05[24+1];
368       TEXT S_DIST_06[24+1];
369       TEXT S_DIST_07[24+1];
370       TEXT S_DIST_08[24+1];
371       TEXT S_DIST_09[24+1];
372       TEXT S_DIST_10[24+1];
373       COUNT S_YTD;
374       COUNT S_ORDER_CNT;
375       COUNT S_REMOTE_CNT;
376       TEXT S_DATA[50+1];
377       } stock_row;
378
379
380   /* Empty field values */
381   #define EMPTY_NUM (MAXINT-1)
382   #define INVALID_NUM (MAXINT)
383   #define EMPTY_FLT   (MAXDOUBLE)
384   #define INVALID_FLT (MINDOUBLE)
385
386   /* Status conditions */
387   #define OK 0
388   #define E 1
389   #define E_INVALID_ITEM 2
390   #define E_NOT_ENOUGH_ORDERS 3
391   #define E_DB_ERROR 4
392
393   /* Error message strings */
394   static char *e_mesg[]={"Transaction com-
 plete.","Error","Invalid item number.",
395                          "Not enough orders.", "Database
 ERROR !!!!"};
396
397
398
399   #define YES 1
400   #define NO 0
401
402
403
404   double cvt_flt();
```

```
405   double cvt_money();
406   TIME getclock();
407   TIME getlocalclock();
408
409   #define TPC_MSG_QUE  150
410
411
412   /*************************************
413   Transaction specific stuff
414   *************************************/
415
416   /* types of transactions */
417   #define NEWORDER 1
418   #define PAYMENT  2
419   #define ORDSTAT  3
420   #define DELIVERY 4
421   #define STOCKLEV 5
422   #define DEFERRED 6   /* deferred portion of delivery
 */
423
424   /* the name of each transaction */
425   static char *transaction_name[] =
426       {"", "New_Order", "Payment", "Order-Status",
427        "Delivery", "Stock-Level", "Deferred-Delivery"};
428
429   /* size of each transaction record */
430   static int transaction_size[] = {0,
431        sizeof(neworder_trans),
432        sizeof(payment_trans),
433        sizeof(ordstat_trans),
434        sizeof(delivery_trans),
435        sizeof(stocklev_trans),
436        sizeof(delivery_trans),
437        0};
438
439   /* valid response time for each transaction */
440   static TIME valid_response[] = {0, 5, 5, 5, 5, 20};
441
442
443   #endif /* TPCC_INCLUDED */
444
```

Appendix B – Database Design

Build

diskinit.sql

```
/*  TPC-C Benchmark Kit                                 */
/*                                                     */
/*  DISKINIT.SQL                                        */
/*                                                     */
/*  6dac 1350 manual      */
/*  This script is used create the database devices */


use master
go

disk init name = "c_log1_dev",
  physname = "x:",
  vdevno   = 14,
  size     = 4096000
go

disk init name = "c_log2_dev",
  physname = "y:",
  vdevno   = 15,
  size     = 4096000
go


disk init name = "c_ordln1_dev",
  physname = "f:",
  vdevno   = 16,
  size     = 3197400
go

disk init name = "c_ordln2_dev",
  physname = "g:",
  vdevno   = 17,
  size     = 3197400
go

disk init name = "c_ordln3_dev",
  physname = "h:",
  vdevno   = 18,
  size     = 3197400
go

disk init name = "c_ordln4_dev",
  physname = "i:",
  vdevno   = 19,
  size     = 3197400
go

disk init name = "c_ordln5_dev",
  physname = "j:",
  vdevno   = 20,
  size     = 3197400
go

disk init name = "c_ordln6_dev",
  physname = "k:",
  vdevno   = 21,
  size     = 3197400
go

disk init name = "c_cs1_dev",
  physname = "l:",
  vdevno   = 22,
  size     = 6726375
go

disk init name = "c_cs2_dev",
  physname = "m:",
  vdevno   = 23,
  size     = 6726375
go

disk init name = "c_cs3_dev",
  physname = "n:",
  vdevno   = 24,
  size     = 6726375
go

disk init name = "c_cs4_dev",
  physname = "o:",
  vdevno   = 25,
  size     = 6726375
go

disk init name = "c_cs5_dev",
  physname = "p:",
  vdevno   = 26,
  size     = 6726375
go

disk init name = "c_cs6_dev",
  physname = "q:",
  vdevno   = 27,
  size     = 6726375
go

disk init name = "c_misc1_dev",
  physname = "r:",
  vdevno   = 28,
  size     = 1154625
go

disk init name = "c_misc2_dev",
  physname = "u:",
  vdevno   = 11,
  size     = 4731000
go
```

createdb.sql

```
/*  TPC-C Benchmark Kit                                 */
/*                                                     */
/*  CREATEDB.SQL                                        */
/*  6dac 1350 manual                                    */
/*  This script is used to create the database    */


use master
go

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

create database tpcc on
        c_ordln1_dev   = 6243,
        c_ordln2_dev   = 6243,
        c_ordln3_dev   = 6243,
        c_ordln4_dev   = 6243,
        c_ordln5_dev   = 6243,
        c_ordln6_dev   = 6243,
        c_cs1_dev    = 9520,
        c_cs2_dev    = 9520,
        c_cs3_dev    = 9520,
        c_cs4_dev    = 9520,
        c_cs5_dev    = 9520,
        c_cs6_dev    = 9520,
        c_cs1_dev    = 2599,
```

```
            c_cs2_dev    = 2599,
            c_cs3_dev    = 2599,
            c_cs4_dev    = 2599,
            c_cs5_dev    = 2599,
            c_cs6_dev    = 2599,
            c_cs1_dev    = 1014,
            c_cs2_dev    = 1014,
            c_cs3_dev    = 1014,
            c_cs4_dev    = 1014,
        c_cs5_dev    = 1014,
            c_cs6_dev    = 1014,
            c_misc1_dev  = 2255,
            c_misc2_dev  = 9235

        log on c_log1_dev =   8000,
            c_log2_dev =  8000
    for load
    go
```

segment.sql

```
    /*  TPC-C Benchmark Kit /
    /* */
    /*  SEGMENT.SQL */
    /*  6dac 1350 manual */
    /*  This script is used to create the database
    segments  */


    use tpcc
    go

    exec sp_addsegment misc_seg, c_misc1_dev
    exec sp_extendsegment misc_seg, c_misc2_dev
    exec sp_addsegment ordln_seg, c_ordln1_dev
    exec sp_extendsegment ordln_seg, c_ordln2_dev
    exec sp_extendsegment ordln_seg, c_ordln3_dev
    exec sp_extendsegment ordln_seg, c_ordln4_dev
    exec sp_extendsegment ordln_seg, c_ordln5_dev
    exec sp_extendsegment ordln_seg, c_ordln6_dev
    exec sp_addsegment cs_seg, c_cs1_dev
    exec sp_extendsegment cs_seg, c_cs2_dev
    exec sp_extendsegment cs_seg, c_cs3_dev
    exec sp_extendsegment cs_seg, c_cs4_dev
    exec sp_extendsegment cs_seg, c_cs5_dev
    exec sp_extendsegment cs_seg, c_cs6_dev
    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_idsmallint,
        w_namechar(10),
        w_street_1char(20),
        w_street_2char(20),
        w_citychar(20),
        w_statechar(2),
        w_zipchar(9),
        w_taxnumeric(4,4),
        w_ytdnumeric(12,2)
    ) on misc_seg
    go
    if exists ( select name from sysobjects where name =
    'district' )
        drop table district
    go
    create table district
    (
        d_idtinyint,
        d_w_idsmallint,
        d_namechar(10),
        d_street_1char(20),
        d_street_2char(20),
        d_citychar(20),
        d_statechar(2),
        d_zipchar(9),
        d_taxnumeric(4,4),
        d_ytdnumeric(12,2),
        d_next_o_idint
    ) on misc_seg
    go
    if exists ( select name from sysobjects where name =
    'customer' )
        drop table customer
    go
    create table customer
    (
        c_idint,
        c_d_idtinyint,
        c_w_idsmallint,
        c_firstchar(16),
        c_middlechar(2),
        c_lastchar(16),
        c_street_1char(20),
        c_street_2char(20),
        c_citychar(20),
        c_statechar(2),
        c_zipchar(9),
        c_phonechar(16),
        c_sincedatetime,
        c_creditchar(2),
        c_credit_limnumeric(12,2),
        c_discountnumeric(4,4),
        c_balancenumeric(12,2),
        c_ytd_paymentnumeric(12,2),
        c_payment_cntsmallint,
        c_delivery_cntsmallint,
        c_data_1char(250),
        c_data_2char(250)
    ) on cs_seg
    go
    if exists ( select name from sysobjects where name =
    'history' )
        drop table history
```

```
go
create table history
(
    h_c_idint,
    h_c_d_idtinyint,
    h_c_w_idsmallint,
    h_d_idtinyint,
    h_w_idsmallint,
    h_datedatetime,
    h_amountnumeric(6,2),
    h_datachar(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_idint,
    no_d_idtinyint,
    no_w_idsmallint
) on misc_seg
go
if exists ( select name from sysobjects where name =
'orders' )
    drop table orders
go
create table orders
(
    o_idint,
    o_d_idtinyint,
    o_w_idsmallint,
    o_c_idint,
    o_entry_ddatetime,
    o_carrier_idtinyint,
    o_ol_cnttinyint,
    o_all_localtinyint
) 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_idint,
    ol_d_idtinyint,
    ol_w_idsmallint,
    ol_numbertinyint,
    ol_i_idint,
    ol_supply_w_idsmallint,
    ol_delivery_ddatetime,
    ol_quantitysmallint,
    ol_amountnumeric(6,2),
    ol_dist_infochar(24)
) on ordln_seg
go
if exists ( select name from sysobjects where name =
'item' )
    drop table item
go
create table item
    (
        i_idint,
        i_im_idint,
        i_namechar(24),
        i_pricenumeric(5,2),
        i_datachar(50)
    ) on misc_seg
    go
    if exists ( select name from sysobjects where name =
    'stock' )
        drop table stock
    go
    create table stock
    (
        s_i_idint,
        s_w_idsmallint,
        s_quantitysmallint,
        s_dist_01char(24),
        s_dist_02char(24),
        s_dist_03char(24),
        s_dist_04char(24),
        s_dist_05char(24),
        s_dist_06char(24),
        s_dist_07char(24),
        s_dist_08char(24),
        s_dist_09char(24),
        s_dist_10char(24),
        s_ytdint,
        s_order_cntsmallint,
        s_remote_cntsmallint,
        s_datachar(50)
    ) on cs_seg
    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
```

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
```

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
```

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 ordln_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
```

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
```

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
```

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
```

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
```

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
```

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
```

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_idint,
             @commit_flag    int,
            @li_id          int,
            @li_s_w_id      smallint,
            @li_qty         smallint,
             @ol_numberint,
             @c_id_localint

    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
             select @s_dist = NULL
                 update stock set s_ytd        = s_ytd +
@li_qty,
                                  s_quantity   = s_quantity
- @li_qty +
                                      case when (s_quantity
- @li_qty < 10) then 91 else 0 end,
                                   @s_quantity  = s_quantity,
                                  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
                 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 val-
ues(@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   */
                                          "jan 1, 1900",
/* 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("%ORIG-
INAL%",@i_data) > 0) and
                                         (patindex("%ORIGI-
NAL%",@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
```

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_idsmallint,
         @d_idtinyint,
         @c_idint,
         @c_lastchar(16) = ""

    as
    declare @c_balancenumeric(12,2),
            @c_firstchar(16),
            @c_middlechar(2),
            @o_idint,
            @o_entry_ddatetime,
            @o_carrier_idsmallint,
            @cntsmallint
    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
```

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
```

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
```

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
```

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
```

tmakefile.x86

```
    !include $(TPC_DIR)\build\ntintel\tpc.inc

    CUR_DIR = $(TPC_DIR)\src

    CLIENT_EXE     = $(EXE_DIR)\client.exe
    MASTER_EXE     = $(EXE_DIR)\master.exe
    TPCCLDR_EXE    = $(EXE_DIR)\tpccldr.exe
    DELIVERY_EXE   = $(EXE_DIR)\delivery.exe
    sqlstat_EXE    = $(EXE_DIR)\sqlstat.exe

    all : $(CLIENT_EXE) $(MASTER_EXE) $(TPCCLDR_EXE)
    $(DELIVERY_EXE) $(sqlstat_EXE)

    $(OBJ_DIR)\client.obj : $(CUR_DIR)\client.c
    $(INC_DIR)\tpcc.h
        $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\client.obj
    $(CUR_DIR)\client.c
```

```
$(OBJ_DIR)\master.obj : $(CUR_DIR)\master.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\master.obj
$(CUR_DIR)\master.c

$(OBJ_DIR)\tpccldr.obj : $(CUR_DIR)\tpccldr.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\tpccldr.obj
$(CUR_DIR)\tpccldr.c

$(OBJ_DIR)\stats.obj : $(CUR_DIR)\stats.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\stats.obj
$(CUR_DIR)\stats.c

$(OBJ_DIR)\getargs.obj : $(CUR_DIR)\getargs.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\getargs.obj
$(CUR_DIR)\getargs.c

$(OBJ_DIR)\util.obj : $(CUR_DIR)\util.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\util.obj
$(CUR_DIR)\util.c

$(OBJ_DIR)\time.obj : $(CUR_DIR)\time.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\time.obj
$(CUR_DIR)\time.c

$(OBJ_DIR)\random.obj : $(CUR_DIR)\random.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\random.obj
$(CUR_DIR)\random.c

$(OBJ_DIR)\strings.obj : $(CUR_DIR)\strings.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\strings.obj
$(CUR_DIR)\strings.c

$(OBJ_DIR)\sqlfuncs.obj : $(CUR_DIR)\sqlfuncs.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\sqlfuncs.obj
$(CUR_DIR)\sqlfuncs.c

$(OBJ_DIR)\tran.obj : $(CUR_DIR)\tran.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\tran.obj
$(CUR_DIR)\tran.c

$(OBJ_DIR)\data.obj : $(CUR_DIR)\data.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\data.obj
$(CUR_DIR)\data.c

$(OBJ_DIR)\delivery.obj : $(CUR_DIR)\delivery.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\delivery.obj
$(CUR_DIR)\delivery.c

$(OBJ_DIR)\sqlstat.obj : $(CUR_DIR)\sqlstat.c
$(INC_DIR)\tpcc.h
    $(CC) $(CFLAGS) /Fo$(OBJ_DIR)\sqlstat.obj
$(CUR_DIR)\sqlstat.c
```

```
$(EXE_DIR)\client.exe : $(OBJ_DIR)\client.obj
$(OBJ_DIR)\tran.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\random.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\data.obj $(OBJ_DIR)\getargs.obj
$(OBJ_DIR)\time.obj $(OBJ_DIR)\stats.obj
$(OBJ_DIR)\strings.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\client.exe               \
    $(OBJ_DIR)\client.obj $(OBJ_DIR)\tran.obj
$(OBJ_DIR)\sqlfuncs.obj   \
    $(OBJ_DIR)\random.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\data.obj       \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj      \

$(OBJ_DIR)\strings.obj
                \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\master.exe : $(OBJ_DIR)\master.obj
$(OBJ_DIR)\sqlfuncs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\master.exe               \
    $(OBJ_DIR)\master.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\util.obj   \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj   \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\tpccldr.exe : $(OBJ_DIR)\tpccldr.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\time.obj $(OBJ_DIR)\random.obj
$(OBJ_DIR)\strings.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\tpccldr.exe              \
    $(OBJ_DIR)\tpccldr.obj $(OBJ_DIR)\getargs.obj
$(OBJ_DIR)\strings.obj \
    $(OBJ_DIR)\util.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\random.obj        \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\delivery.exe : $(OBJ_DIR)\delivery.obj
$(OBJ_DIR)\sqlfuncs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\delivery.exe                 \
    $(OBJ_DIR)\delivery.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\util.obj      \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj      \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)

$(EXE_DIR)\sqlstat.exe : $(OBJ_DIR)\sqlstat.obj
$(OBJ_DIR)\sqlfuncs.obj $(OBJ_DIR)\util.obj
$(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj
    $(LL) -entry:mainCRTStartup -
out:$(EXE_DIR)\sqlstat.exe                \
    $(OBJ_DIR)\sqlstat.obj $(OBJ_DIR)\sqlfuncs.obj
$(OBJ_DIR)\util.obj      \
    $(OBJ_DIR)\getargs.obj $(OBJ_DIR)\time.obj
$(OBJ_DIR)\stats.obj      \
    $(DB_LIB)\ntwdblib.lib $(NTLIBS)
```

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;
}
```

strings.c

```
/*  FILE:STRINGS.C
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96, By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:String generation functions for
Microsoft TPC-C Benchmark Kit
 *  Author:Damien Lindauer
 *  damienl@Microsoft.com
 */
// Includes
#include "tpcc.h"
#include <string.h>
#include <ctype.h>
//================================================
====================
//
// Function name: MakeAddress
//
//================================================
====================
void MakeAddress(char *street_1,
     char *street_2,
     char *city,
      char *state,
      char *zip)
{
```

```
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAddress()\n",
(int) GetCurrentThreadId());
#endif
    MakeAlphaString (10, 20, ADDRESS_LEN, street_1);
    MakeAlphaString (10, 20, ADDRESS_LEN, street_2);
    MakeAlphaString (10, 20, ADDRESS_LEN, city);
    MakeAlphaString ( 2,  2, STATE_LEN, state);
    MakeZipNumberString( 9,  9, ZIP_LEN, zip);
#ifdef DEBUG
    printf("[%ld]DBG: MakeAddress: street_1: %s,
street_2: %s, city: %s, state: %s, zip: %s\n",
    (int) GetCurrentThreadId(), street_1, street_2,
city, state, zip);
#endif
    return;
}
//=================================================
==================
//
// Function name: LastName
//
//=================================================
==================
void LastName(int  num,
          char *name)
{
    inti;
    intlen;
    static char *n[] =
    {
    "BAR" , "OUGHT", "ABLE" , "PRI"  , "PRES",
    "ESE" , "ANTI" , "CALLY", "ATION", "EING"
    };
#ifdef DEBUG
    printf("[%ld]DBG: Entering LastName()\n", (int)
GetCurrentThreadId());
#endif
    if ((num >= 0) && (num < 1000))
    {
    strcpy(name, n[(num/100)%10]);
    strcat(name, n[(num/10)%10]);
    strcat(name, n[(num/1)%10]);

    if (strlen(name) < LAST_NAME_LEN)
    {
    PaddString(LAST_NAME_LEN, name);
    }
    }
    else
    {
    printf("\nError in LastName()... num <%ld> out
of range (0,999)\n", num);
    exit(-1);
    }

#ifdef DEBUG
    printf("[%ld]DBG: LastName: num = [%d] ==>
[%d][%d][%d]\n",
    (int) GetCurrentThreadId(), num, num/100,
(num/10)%10, num%10);
    printf("[%ld]DBG: LastName: String = %s\n",
(int) GetCurrentThreadId(), name);
#endif
    return;
}
//=================================================
==================
//
// Function name: MakeAlphaString
//
//=================================================
==================
//philipdu 08/13/96 Changed MakeAlphaString to use A-
Z, a-z, and 0-9 in
//accordance with spec see below:
//The spec says:
//4.3.2.2The notation random a-string [x .. y]
//(respectively, n-string [x .. y]) represents a
string of random alphanumeric
//(respectively, numeric) characters of a random
length of minimum x, maximum y,
//and mean (y+x)/2. Alphanumerics are A..Z, a..z,
and 0..9.  The only other
//requirement is that the character set used "must
be able to represent a minimum
//of 128 different characters".  We are using 8-bit
chars, so this is a non issue.
//It is completely unreasonable to stuff non-
printing chars into the text fields.
//-CLevine 08/13/96
int MakeAlphaString( int  x, int  y, int  z, char
*str)
{
    intlen;
    inti;
    staticchar chArray[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop
qrstuvwxyz";
    staticintchArrayMax = 61;
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAlphaString()\n",
(int) GetCurrentThreadId());
#endif
    len= RandomNumber(x, y);
    for (i=0; i<len; i++)
    str[i] = chArray[RandomNumber(0, chArrayMax)];
    if ( len < z )
    memset(str+len, ' ', z - len);
    str[len] = 0;

    return len;
}
#if 0
//philipdu 08/13/96 Orginal MakeAlphaString
int MakeAlphaString( int  x,
         int  y,
     int  z,
         char *str)
{

    intlen;
    inti;
#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAlphaString()\n",
(int) GetCurrentThreadId());
#endif
    len= RandomNumber(x, y);
    for (i=0; i<len; i++)
```

```
    {
    str[i] = RandomNumber(MINPRINTASCII,
MAXPRINTASCII);
    }
    str[len] = '\0';

    if (len < z)
    {
    PaddString(z, str);
    }
    return (len);
}
#endif
//=================================================
==================
//
// Function name: MakeOriginalAlphaString
//
//=================================================
==================
int MakeOriginalAlphaString(int  x,
        int  y,
    int  z,
        char *str,
        int  percent)
{
    intlen;
    intval;
    intstart;
#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeOriginalAlphaString()\n", (int)
GetCurrentThreadId());
#endif
    // verify prercentage is valid
    if ((percent < 0) || (percent > 100))
    {
    printf("MakeOrigianlAlphaString: Invalid
percentage: %d\n", percent);
    exit(-1);
    }
    // verify string is at least 8 chars in length
    if ((x + y) <= 8)
    {
    printf("MakeOriginalAlphaString: string length
must be >= 8\n");
    exit(-1);
    }
    // Make Alpha String
    len = MakeAlphaString(x,y, z, str);
    val = RandomNumber(1,100);
    if (val <= percent)
    {
    start = RandomNumber(0, len - 8);
    strncpy(str + start, "ORIGINAL", 8);
    }

#ifdef DEBUG
    printf("[%ld]DBG: MakeOriginalAlphaString: :
%s\n",
    (int) GetCurrentThreadId(), str);
#endif
    return strlen(str);
}

//=================================================
==================
//
// Function name: MakeNumberString
//
//=================================================
==================
int MakeNumberString(int  x, int  y, int  z, char
*str)
{
    char tmp[16];
    //MakeNumberString is always called
MakeZipNumberString(16, 16, 16, string)
    memset(str, '0', 16);
    itoa(RandomNumber(0, 99999999), tmp, 10);
    memcpy(str, tmp, strlen(tmp));
    itoa(RandomNumber(0, 99999999), tmp, 10);
    memcpy(str+8, tmp, strlen(tmp));
    str[16] = 0;
    return 16;
}
#if 0
int MakeNumberString(int  x,
        int  y,
     int  z,
        char *str)
{
    intlen;
    inti;
#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeNumberString()\n", (int) GetCurrentThreadId());
#endif
    len = RandomNumber(x,y);
    for (i=0; i < len; i++)
    {
    str[i] = (char) (RandomNumber(48,57));
    }

    str[len] = '\0';
    PaddString(z, str);
    return strlen(str);
}
#endif
//=================================================
==================
//
// Function name: MakeZipNumberString
//
//=================================================
==================
int MakeZipNumberString(int  x, int  y, int  z, char
*str)
{
    char tmp[16];
    //MakeZipNumberString is always called
MakeZipNumberString(9, 9, 9, string)
    strcpy(str, "000011111");
    itoa(RandomNumber(0, 9999), tmp, 10);
    memcpy(str, tmp, strlen(tmp));
    return 9;
}
#if 0
//pgd 08/14/96 Orginal Code Below
```

```
int MakeZipNumberString(int  x,
        int  y,
     int  z,
         char *str)
{
    intlen;
    inti;
#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeZipNumberString()\n", (int)
GetCurrentThreadId());
#endif
    len = RandomNumber(x-5,y-5);
    for (i=0; i < len; i++)
    {
    str[i] = (char) (RandomNumber(48,57));
    }

    str[len] = '\0';
    strcat(str, "11111");
    PaddString(z, str);
    return strlen(str);
}
#endif
//=================================================
==================
//
// Function name: InitString
//
//=================================================
==================
void InitString(char *str, int len)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitString()\n",
(int) GetCurrentThreadId());
#endif
    memset(str, ' ', len);
    str[len] = 0;
}
#if 0
//Orginal pgd 08/14/96
void InitString(char *str, int len)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitString()\n",
(int) GetCurrentThreadId());
#endif
    for (i=0; i< len; i++)
    str[i] = ' ';
    str[len] = '\0';
}
#endif
//=================================================
==================
// Function name: InitAddress
//
// Description:
//
//=================================================
==================
void InitAddress(char *street_1, char *street_2,
char *city, char *state, char *zip)
{
    int i;
    memset(street_1, ' ', ADDRESS_LEN+1);
    memset(street_2, ' ', ADDRESS_LEN+1);
    memset(city, ' ', ADDRESS_LEN+1);
    street_1[ADDRESS_LEN+1] = 0;
    street_2[ADDRESS_LEN+1] = 0;
    city[ADDRESS_LEN+1]     = 0;
    memset(state, ' ', STATE_LEN+1);
    state[STATE_LEN+1] = 0;
    memset(zip, ' ', ZIP_LEN+1);
    zip[ZIP_LEN+1] = 0;
}
#if 0
//Orginal pgd 08/14/96
void InitAddress(char *street_1,
     char *street_2,
      char *city,
      char *state,
      char *zip)
{
    int i;
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitAddress()\n",
(int) GetCurrentThreadId());
#endif
    for (i=0; i< ADDRESS_LEN+1; i++)
    {
    street_1[i] = ' ';
    street_2[i] = ' ';
    city[i]     = ' ';
    }
    street_1[ADDRESS_LEN+1] = '\0';
    street_2[ADDRESS_LEN+1] = '\0';
    city[ADDRESS_LEN+1]     = '\0';
    for (i=0; i< STATE_LEN+1; i++)
    state[i] = ' ';
    state[STATE_LEN+1] = '\0';
    for (i=0; i< ZIP_LEN+1; i++)
    zip[i] = ' ';
    zip[ZIP_LEN+1] = '\0';
}
#endif
//=================================================
==================
//
// Function name: PaddString
//
//=================================================
==================
void PaddString(int max, char *name)
{
    inti;
    intlen;
    len = strlen(name);
    if ( len < max )
    memset(name+len, ' ', max - len);
    name[max] = 0;
    return;
}
#if 0
```

```
        //pgd 08/14/96 Orginal code below
        void PaddString(intmax,
        char*name)
        {
        inti;
        intlen;
        #ifdef DEBUG
        printf("[%ld]DBG: Entering PaddString()\n",
    (int) GetCurrentThreadId());
        #endif
        len = strlen(name);
        for (i=1;i<=(max - len);i++)
        {
        strcat(name, " ");
        }
        }
    #endif
```

time.c

```
    // TPC-C Benchmark Kit
    //
    // Module:  TIME.C
    // Author:  DamienL
    // Includes
    #include "tpcc.h"
    // Globals
    static long start_sec;
    //=================================================
    ====================
    //
    // Function name: TimeNow
    //
    //=================================================
    ====================
    long TimeNow()
    {
       longtime_now;
       struct_timeb el_time;
    #ifdef DEBUG
        printf("[%ld]DBG: Entering TimeNow()\n", (int)
    GetCurrentThreadId());
    #endif
       _ftime(&el_time);
       time_now = ((el_time.time - start_sec) * 1000) +
    el_time.millitm;
       return time_now;
    }
    //=================================================
    ====================
    //
    // Function name: TimeInit
    //
    // This function is used to normalize the seconds
    component of
    // elapsed time so that it will not overflow, when
    converted to milli seconds
    //
    //=================================================
    ====================
    void TimeInit()
    {
        struct  _timeb  norm_time;
    #ifdef DEBUG
        printf("[%ld]DBG: Entering TimeInit()\n", (int)
    GetCurrentThreadId());
    #endif
        _ftime(&norm_time);
        start_sec = norm_time.time;
    }
    //=================================================
    ====================
    //
    // Function name: TimeKeying
    //
    //=================================================
    ====================
    void TimeKeying(intTranType,
        doubleload_multiplier)
    {
    #ifdef DEBUG
        printf("[%ld]DBG: Entering TimeKeying()\n",
    (int) GetCurrentThreadId());
    #endif
        switch (TranType)
        {
            case NEW_ORDER_TRAN:
        UtilSleepMs( (long) ((load_multiplier *
    18)*1000) );
        break;
            case PAYMENT_TRAN:
        UtilSleepMs( (long) ((load_multiplier * 3)*1000)
    );
        break;
            case ORDER_STATUS_TRAN:
            case DELIVERY_TRAN:
            case STOCK_LEVEL_TRAN:
        UtilSleepMs( (long) ((load_multiplier * 2)*1000)
    );
        break;
            default:
        printf("TimeKeying: Error - default reached!\n");
        }
    }
    //=================================================
    ====================
    //
    // Function name: TimeThink
    //
    //=================================================
    ====================
    void TimeThink(intTranType,
           doubleload_multiplier)
    {
    #ifdef DEBUG
        printf("[%ld]DBG: Entering TimeThink()\n", (int)
    GetCurrentThreadId());
    #endif
        switch (TranType)
        {
            case NEW_ORDER_TRAN:
            case PAYMENT_TRAN:
        UtilSleepMs( (long) ((load_multiplier *
    12)*1000) );
        break;
            case ORDER_STATUS_TRAN:
        UtilSleepMs( (long) ((load_multiplier *
    10)*1000) );
```

```
        break;
            case DELIVERY_TRAN:
            case STOCK_LEVEL_TRAN:
        UtilSleepMs( (long) ((load_multiplier * 5)*1000)
    );
        break;
            default:
        printf("TimeThink: Error - default reached!\n");
        }
    }
```

tpcc.h

```
    /*  FILE:TPCC.H
     *  Microsoft TPC-C Kit Ver. 3.00.000
     *  Audited 08/23/96, By Francois Raab
     *
     *  Copyright Microsoft, 1996
     *
     *  PURPOSE:Header file for Microsoft TPC-C
    Benchmark Kit
     *  Author:Damien Lindauer
     *  damienl@Microsoft.com
     */
    // Build number of TPC Benchmark Kit
    #define TPCKIT_VER "3.00.00"
    // General headers
    #include <windows.h>
    #include <winbase.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <process.h>
    #include <stddef.h>
    #include <stdarg.h>
    #include <string.h>
    #include <signal.h>
    #include <time.h>
    #include <timeb.h>
    #include <types.h>
    #include <wincon.h>
    #ifdef USE_ODBC
    // ODBC headers
    #include <sql.h>
    #include <sqlext.h>
    HENV henv;
    #endif
    // DB-Library headers
    #include <sqlfront.h>
    #include <sqldb.h>
    #include "trans.h"//pgd 5-6-96 split transaction
    structs definations into own header
        //for tpcform.c i.e. telnet application
    // Critical section declarations
    CRITICAL_SECTIONConsoleCritSec;
    CRITICAL_SECTIONQueuedDeliveryCritSec;
    CRITICAL_SECTIONWriteDeliveryCritSec;
    CRITICAL_SECTIONDroppedConnectionsCritSec;
    CRITICAL_SECTIONClientErrorLogCritSec;
    // General constants
    #define SQLCONN DBPROCESS
    #define DUMB_MESSAGE     5701
    #define ABORT_ERROR      6104
    #define INVALID_ITEM_ID 0
    #define MILLI            1000
    #define MAX_THREADS      2510
    #define STATS_MSG_LOW   3600
    #define STATS_MSG_HIGH  3700
    #define SHOWPLAN_MSG_LOW   6200
    #define SHOWPLAN_MSG_HIGH  6300
    #define FALSE           0
    #define TRUE            1
    #define UNDEF   -1
    #define MINPRINTASCII   32
    #define MAXPRINTASCII  126
    // Default environment constants
    #define SERVER          ""
    #define DATABASE        "tpcc"
    #define USER            "sa"
    #define PASSWORD            ""
    #define SYNCH_SERVERNAME""
    // Statistic constants
    #define INTERVAL           20    // Total interval of
    buckets, in sec
    #define UNIT              .1  // Time period of each bucket
    #define HIST_MAX         200  // Num of histogram buckets
    = INTERVAL/UNIT
    #define BUCKET            100   // Division factor for
    response time
    // Default master arguments
    #define ADMIN_DATABASE        "tpcc_admin"
    #define RAMP_UP               600
    #define STEADY_STATE         1200
    #define RAMP_DOWN             120
    #define NUM_USERS              10
    #define NUM_WAREHOUSES      1
    #define THINK_TIMES0
    #define DISPLAY_DATA        0
    #define DEFMSPACKSIZE     4096
    #define TRANSACTION         0
    #define CLIENT_MODE           1
    #define DEF_WW_T  120
    #define DEF_WW_a1
    #define DEADLOCK_RETRY         4
    #define DELIVERY_BACKOFF2
    #define DELIVERY_MODE0
    #define NEWORDER_MODE0
    #define DEF_LOAD_MULTIPLIER   1.0
    #define DEF_CHECKPOINT_INTERVAL   960
    #define DEF_FIRST_CHECKPOINT  240
    #define DISABLE_90TH0
    #defineRESFILENAME"results.txt"
    #define SQLSTAT_FILENAME"sqlstats.txt"
    #define ENABLE_SQLSTAT0
    #define SQLSTAT_PERIOD  100
    #define SHUTDOWN_SERVER0
    #define AUTO_RUN0
    #define DISABLE_SQLPERF0
    // Default client arguments
    #define NUM_THREADS       10
    #define X_FLAG                      0
    #define Y_FLAG                      1
    #define NUM_DELIVERIES2
    #define CLIENT_NURAND      223
    #define DISABLE_DELIVERY_RESFILES   1
    #define ENABLE_QJ0
```

```
// Globals for queued delivery handling
typedef struct delivery_node *DELIVERY_PTR;
DELIVERY_PTRdelivery_head, delivery_tail;
shortqueued_delivery_cnt;
HANDLEhDeliveryMonPipe;
struct delivery_node
{
    shortw_id;
    shorto_carrier_id;
    SYSTEMTIMEqueue_time;
    longtran_start_time;
    structdelivery_node *next_delivery;
};
// Default loader arguments
#define BATCH           10000
#define DEFLDPACKSIZE 4096
#define ORDERS_PER_DIST  3000
#define LOADER_RES_FILE     "load.out"
#define LOADER_NURAND_C  123
#define DEF_STARTING_WAREHOUSE1
#define BUILD_INDEX1
#define INDEX_SCRIPT_PATH"scripts"
// Transaction types
#define EMPTY               0
#define NEW_ORDER_TRAN      1
#define PAYMENT_TRAN        2
#define ORDER_STATUS_TRAN   3
#define DELIVERY_TRAN       4
#define STOCK_LEVEL_TRAN    5
// Statistic structures
typedef struct
{
    long   tran_count;
    long   total_time;
    long   resp_time;
    long   resp_min;
    long   resp_max;
    long   rolled_back;
    long   tran_2sec;
    long   tran_5sec;
    long   tran_sqr;
    long   num_deadlocks;
    long   resp_hist[HIST_MAX];
} TRAN_STATS;
typedef struct
{
    TRAN_STATS NewOrderStats;
    TRAN_STATS PaymentStats;
    TRAN_STATS OrderStatusStats;
    TRAN_STATS QueuedDeliveryStats;
    TRAN_STATSDeliveryStats;
    TRAN_STATS StockLevelStats;
} CLIENT_STATS;
// driver structures
typedef struct
{
    char  *server;
    char  *database;
    char  *user;
    char  *password;
    char  *table;
    long   num_warehouses;
    long   batch;
    long   verbose;
    long   pack_size;
    char*loader_res_file;
    char*synch_servername;
    longcase_sensitivity;
    longstarting_warehouse;
    longbuild_index;
    char*index_script_path;
} TPCCLDR_ARGS;
typedef struct
{
    char  *server;
    char  *user;
    char  *password;
    char  *admin_database;
    char  *sqlstat_filename;
    longrun_id;
} SQLSTAT_ARGS;
typedef struct
{
    SQLCONN  *sqlconn;
    char    *server;
    char     *database;
    char*admin_database;
    char     *user;
    char     *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long      num_users;
    long      num_warehouses;
    long      think_times;
    long      display_data;
    long     client_mode;
    long     tran;
    longdeadlock_retry;
    longdelivery_backoff;
    longnum_deliveries;
    char *comment;
    doubleload_multiplier;
    longcheckpoint_interval;
    longfirst_checkpoint;
    longdisable_90th;
    char*resfilename;
    char*sqlstat_filename;
    longenable_sqlstat;
    longsqlstat_period;
    longshutdown_server;
    longauto_run;
    longdropped_connections;
    short     spid;
    longdisable_sqlperf;
} MASTER_DATA;
typedef struct
{
    long  num_threads;
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
```

```
    long      pack_size;
    shortx_flag;
    char*synch_servername;
    longdisable_delivery_resfiles;
    longenable_qj;
#ifdef USE_CONMON
    HANDLE  hConMon;
    short  con_id;
    short  con_x;
    short  con_y;
#endif
} GLOBAL_CLIENT_DATA;
typedef struct
{
#ifdef USE_ODBC
    HDBChdbc;
    HSTMThstmt;
#else
    SQLCONN   *sqlconn;
#endif
    short  threadid;
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long     num_warehouses;
    long     client_mode;
    long     tran;
    longdeadlock_retry;
    long      think_times;
    long      pack_size;
    long      tran_start_time;
    long      tran_end_time;
    long      display_data;
    long     id;
    short     w_id;
    short     spid;
    longdisable_90th;
    doubleload_multiplier;
    longnum_deliveries;
    longenable_qj;
#ifdef USE_CONMON
    HANDLE  hConMon;
    short  con_id;
    short  con_x;
    short  con_y;
    shortfTimerStat;
#endif
} CLIENT_DATA;
typedef struct
{
#ifdef USE_ODBC
    HDBChdbc;
    HSTMThstmt;
#else
    SQLCONN   *sqlconn;
#endif
    SYSTEMTIMEqueue_time;
    SYSTEMTIMEcompletion_time;
    long      tran_start_time;
    long      tran_end_time;
    short  threadid;
    FILE      *fDelivery;
    short     spid;
    short     w_id;
    shortd_id;
    short     o_carrier_id;
    DEL_ITEM  DelItems[10];
    char      *server;
    char      *database;
    char*admin_database;
    char      *user;
    char      *password;
    long      ramp_up;
    long      steady_state;
    long      ramp_down;
    long      pack_size;
    long     id;
    longdisable_90th;
    longdelivery_backoff;
    longdisable_delivery_resfiles;
    longenable_qj;
} DELIVERY;
typedef struct
{
    longpipe_num;
} DELIVERY_ARGS;
// For client synchronization
#define LINE_LEN   80
#define NAME_SIZE  25
#define IN_BUF_SIZE        1000
#define OUT_BUF_SIZE       1000
#define TIME_OUT           0
#define PLEASE_READ        1000
#define PLEASE_WRITE       1000
typedef struct _WRTHANDLE
{   HANDLEhPipe;
    DWORDthreadID;
    CHARName[NAME_SIZE];
    struct _WRTHANDLE *next;
}WRTHANDLE;
// For client console monitor
#ifdef USE_CONMON
#defineCON_LINE_SIZE40
#defineDEADLOCK_X17
#define DEADLOCK_Y4
#define CUR_STATE_X15
#define CUR_STATE_Y3
#defineYELLOW0
#defineRED1
#defineGREEN2
int total_deadlocks;
#endif
// Functions in random.c
void   seed();
long   irand();
doubledrand();
voidWUCreate();
shortWURand();
// Functions in getargs.c;
```

```
void GetArgsLoader();
void GetArgsLoaderUsage();
void GetArgsMaster();
void GetArgsMasterUsage();
void GetArgsClient();
void GetArgsClientUsage();
void GetArgsDelivery();
void GetArgsDeliveryUsage();
void GetArgsSQLStat();
void GetArgsSQLStatUsage();
// Functions in master.c
void ReadClientDone();
BOOL    CtrlHandler();
// Functions in client.c
void  ClientMain();
voidDeliveryMain();
voidDelivery();
void  ClientEmulate();
short ClientSelectTransaction();
void  ClientShuffleDeck();
//Functions in tran.c
BOOL TranNewOrder();
BOOL TranPayment();
BOOL TranOrderStatus();
BOOL TranDelivery();
BOOL TranStockLevel();
// Functions in data.c
void DataNewOrder();
void DataPayment();
void DataOrderStatus();
void DataDelivery();
void DataStockLevel();
short DataRemoteWarehouse();
// Functions in time.c
long TimeNow();
void TimeInit();
void TimeKeying();
void TimeThink();
// Functions in stats.c
void StatsInit();
void StatsInitTran();
void StatsGeneral();
void StatsDelivery();
// Functions in sqlfuncs.c
BOOL  SQLExec();
BOOL  SQLExecCmd();
BOOL  SQLOpenConnection();
void  SQLClientInit();
int   SQLMasterInit();
voidSQLDeliveryInit();
int   SQLClientStats();
int   SQLDeliveryStats();
void  SQLTranStats();
void  SQLMasterStats();
void  SQLMasterTranStats();
void  SQLIOStats();
void  SQLCheckpointStats();
voidSQLInitResFile();
void  SQLGetRunId();
BOOL  SQLNewOrder();
BOOL  SQLPayment();
BOOL  SQLOrderStatus();
BOOLSQLStockLevel();
void    SQLDelivery();
int   SQLGetCustId();
void   SQLExit();
void  SQLInit();
void  SQLInitPrivate();
void  SQLClientInitPrivate();
void  SQLDeliveryInitPrivate();
int   SQLMsgHandler();
int   SQLErrHandler();
int   SQLClientMsgHandler();
int   SQLClientErrHandler();
int   SQLDeliveryMsgHandler();
int   SQLDeliveryErrHandler();
void  SQLInitDate();
voidSQLShutdown();
#ifdef USE_ODBC
void ODBCOpenConnection();
void ODBCOpenDeliveryConnection();
BOOLODBCError();
voidODBCExit();
#endif
// Functions in util.c
void UtilSleep();
void UtilPrintNewOrder();
void UtilPrintPayment();
void UtilPrintOrderStatus();
void UtilPrintDelivery();
void UtilPrintStockLevel();
void UtilPrintOlTable();
void UtilError();
void UtilFatalError();
void UtilStrCpy();
#ifdef USE_CONMON
void WriteConsoleString();
#endif
voidWriteDeliveryString();
BOOLAddDeliveryQueueNode();
BOOLGetDeliveryQueueNode();
// Functions in strings.c
void MakeAddress();
void LastName();
int  MakeAlphaString();
int  MakeOriginalAlphaString();
int  MakeNumberString();
int  MakeZipNumberString();
void InitString();
void InitAddress();
void PaddString();
// Functions in delivery.c
void  DeliveryHMain();
void DeliveryH();
```

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
    charol_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
{
    longc_id;
    shortc_d_id;
    shortc_w_id;
    charc_first[FIRST_NAME_LEN+1];
    charc_middle[MIDDLE_NAME_LEN+1];
    charc_last[LAST_NAME_LEN+1];
    charc_street_1[ADDRESS_LEN+1];
    charc_street_2[ADDRESS_LEN+1];
    charc_city[ADDRESS_LEN+1];
    charc_state[STATE_LEN+1];
    charc_zip[ZIP_LEN+1];
    charc_phone[PHONE_LEN+1];
    charc_credit[CREDIT_LEN+1];
    doublec_credit_lim;
    doublec_discount;
    doublec_balance;
    doublec_ytd_payment;
    shortc_payment_cnt;
    shortc_delivery_cnt;
    charc_data_1[C_DATA_LEN+1];
    charc_data_2[C_DATA_LEN+1];
    doubleh_amount;
    charh_data[H_DATA_LEN+1];
} CUSTOMER_STRUCT;
typedef struct
{
    charc_last[LAST_NAME_LEN+1];
    charc_first[FIRST_NAME_LEN+1];
    longc_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;
    RETCODEretcode;
    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];
    charbuf[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];
    charc_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];
    charh_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;
    intj;
    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;
    charo_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];
    charol_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_deliver
y_d);

strcpy(ol_dist_info,orders_buf[i].o_ol[j].ol_dist_inf
o);
    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)
{

    longtime_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()
{
    RETCODEretcode;
    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)
{
    charcmd[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);
}
```

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",
    (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",
        (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"
           "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_PTRlocal_node;
#ifdef DEBUG
    DELIVERY_PTRptrtmp;
    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_PTRlocal_node;
    BOOLrc;
#ifdef DEBUG
    DELIVERY_PTRptrtmp;
    shorti;
#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);
}
```

tpc.inc

```
#
#########################################################
#######################
#       TPC.INC
#
#
#########################################################
#######################
# TYPE will be supplied as the type directory.
EXE_DIR = $(TPC_DIR)\run\ntintel
OBJ_DIR = $(TPC_DIR)\build\ntintel\obj
INC_DIR = $(TPC_DIR)\src
```

```
# C compiler flags.
# NT_WIN32 is always small model.
# OF will be supplied as the optimizing flag (/Od or
/Ot).
# ZF will be supplied as the debugging flag (none or
/Zi).
# DB will be supplied as a debugging flag.
CDEFINES = -DWIN32  -DNTWIN32 -Di386 -DDBNTWIN32  -
D_X86_ -DCONSOLE -D_WINDOWS -D_NTWIN
CFLAGS = /c /G4 /Gs $(OF) /W2 $(ZF) $(DB) $(DBAPI)
$(CDEFINES) /DLINT_ARGS=1
CFLAGSOPT = $(CFLAGS)  /Ot
CC = cl
# Linker flags.
# LF1 will be supplied as the link debugging flag (-
debug:full)
# LF2 will be supplied as the link debugging flag (-
debugtype:cv)
LFLAGS = -subsystem:console $(LF1) $(LF2)
/NODEFAULTLIB:LIBC
LL = link $(LFLAGS)
# NTWIN32 libraries
# BUGBUG: Can't load strings in console subsystem
mode yet.
NTLIBS= $(NTLIB)\kernel32.lib \
    $(NTLIB)\advapi32.lib \
    $(NTLIB)\libcmt.lib
```

Appendix C – Tunable Parameters

Microsoft Windows NT Version 4.0 Configuration Parameters

The following services were disabled in the Windows NT Control Panel/Services:

Alerter
Computer Browser
Liscense Logging Service
Network DDE
Schedule
Net Logon
Messenger
NT LM Security Support Provider
Plug and Plan
Spooler
TCP/IP Netbios Helper

Server System Configuration Parameters

```
Microsoft Diagnostics Report For \\PRF_SUT6
------------------------------------------------------------
-------------

OS Version Report
------------------------------------------------------------
-------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiproces-
sor Free
Registered Owner: HP, NSD
Product Number: 50382-270-8019154-83617
------------------------------------------------------------
-------------

System Report
------------------------------------------------------------
-------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 12/26/97
BIOS Version: <unavailable>

Processor list:
   0:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   1:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   2:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   3:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   4:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   5:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   6:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
   7:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~198
Mhz
------------------------------------------------------------
-------------

Video Display Report
------------------------------------------------------------
-------------
BIOS Date: 05/22/96
BIOS Version: CL-GD5436/46 PCI VGA BIOS Version 1.25

Adapter:
   Setting: 1024 x 768 x 256
            70 Hz
   Type: cirrus compatible display adapter
   String: Cirrus Logic Compatible
   Memory: 2 MB
   Chip Type: Cirrus Logic 5446
   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)  Total: 7,831,684KB, Free: 2,806,352KB
   Serial Number: 2CDD - 9F3A
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
D:\  (Local - FAT)  Total: 1,052,064KB, Free: 896,544KB
   Serial Number: 7833 - 281E
   Bytes per cluster: 512
   Sectors per cluster: 64
   Filename length: 255
E:\  (Local - NTFS)  Total: 136,469,484KB, Free:
35,597,136KB
   Serial Number: 443A - A65B
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
V:\  (Local - NTFS)  Total: 136,469,484KB, Free:
35,597,136KB
   Serial Number: 5C68 - 206A
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255

Memory Report
------------------------------------------------------------
-------------
Handles: 1,764
Threads: 148
Processes: 27

Physical Memory (K)
   Total: 4,095,404
   Available: 3,914,712
   File Cache: 14,100

Kernel Memory (K)
   Total: 336,836
   Paged: 81,420
   Nonpaged: 255,416

Commit Charge (K)
   Total: 114,732
   Limit: 8,128,860
   Peak: 3,204,384

Pagefile Space (K)
   Total: 4,193,280
   Total in use: 88,492
   Peak: 1,293,956
```

```
   C:\pagefile.sys
      Total: 4,193,280
      Total in use: 88,492
      Peak: 1,293,956


Services Report
--------------------------------------------------------
-------------
Alerter                                        Stopped
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
Ataman TCP Remote Logon Services               Running
(Automatic)
   C:\atrls2\ATRLS.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Computer Browser                               Stopped
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
      LmHosts
ClipBook Server                                Stopped
(Manual)
   C:\WINNT\system32\clipsrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      NetDDE
DHCP Client (TDI)                              Stopped
(Disabled)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Tcpip
      Afd
      NetBT
EventLog (Event log)                           Running
(Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Gopher Publishing Service                      Stopped
(Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
Server                                         Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
Workstation (NetworkProvider)                  Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
License Logging Service                        Stopped
(Automatic)
   C:\WINNT\System32\llssrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TCP/IP NetBIOS Helper                          Stopped
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      NetworkProvider
Messenger                                      Stopped
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      NetBios
Monitor Service                                Stopped
(Manual)
   C:\WINNT\system32\datalog.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
MSDTC (MS Transactions)                        Stopped
(Manual)
   C:\MSSQL\BINN\msdtc.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      RPCSS
FTP Publishing Service                         Stopped
(Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
MSSQLServer                                 Stopped   (Man-
ual)
   C:\MSSQL\BINN\SQLSERVR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
Network DDE (NetDDEGroup)                      Stopped
(Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      NetDDEDSDM
Network DDE DSDM                               Stopped
(Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Net Logon (RemoteValidation)                   Stopped
(Manual)
   C:\WINNT\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)
   C:\WINNT\System32\SERVICES.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Plug and Play (PlugPlay)                      Stopped
(Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Directory Replicator                          Stopped
(Manual)
   C:\WINNT\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)
   C:\WINNT\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)
   C:\WINNT\system32\RpcSs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Schedule                                   Stopped   (Man-
ual)
   C:\WINNT\System32\AtSvc.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Spooler (SpoolerGroup)                        Stopped
(Automatic)
   C:\WINNT\system32\spoolss.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
SQLExecutive                               Stopped   (Man-
ual)
   C:\MSSQL\BINN\SQLEXEC.EXE
   Service Account Name: .\Administrator
   Error Severity: Normal
   Service Flags: Own Process
Telephony Service                             Stopped
(Manual)
   C:\WINNT\system32\tapisrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
UPS                                        Stopped   (Man-
ual)
   C:\WINNT\System32\ups.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
World Wide Web Publishing Service             Stopped
(Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP



Drivers Report
--------------------------------------------------------
-------------
Abiosdsk (Primary disk)                       Stopped
(Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
AFD Networking Support Environment (TDI)      Running
(Automatic)
   C:\WINNT\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)                       Stopped
(Disabled)
   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
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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Beep (Base)                                Running   (Sys-
tem)
   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   (Sys-
tem)
   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)                        Running
(Boot)
   C:\WINNT\System32\drivers\dac960nt.sys
   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
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
et4000 (Video)                              Stopped   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Fastfat (Boot file system)                      Running
(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
HP 10/100TX PCI Ethernet Adapter Driver (NDIS) Running
(Automatic)
   C:\WINNT\System32\drivers\hptx.sys
   Error Severity: Normal
   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   (Sys-
tem)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mga (Video)                                   Stopped   (Dis-
abled)
   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
Msfs (File system)                              Running
(System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                                 Running   (Man-
ual)
   C:\WINNT\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)                Running
(Manual)
   C:\WINNT\System32\drivers\netbios.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      TDI
WINS Client(TCP/IP) (PNP_TDI)                   Running
(Automatic)
   C:\WINNT\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                                    Stopped   (Man-
ual)
   C:\WINNT\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   (Sys-
tem)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
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
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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Rdr (Network)                                Running   (Man-
ual)
   C:\WINNT\System32\drivers\rdr.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
s3 (Video)                                   Stopped   (Dis-
abled)
   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)                           Running
(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   (Man-
ual)
```

```
   C:\WINNT\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
symc8XX (SCSI miniport)                         Running
(Boot)
   C:\WINNT\system32\drivers\symc8XX.sys
   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)
   C:\WINNT\System32\drivers\tcpip.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
tga (Video)                                   Stopped   (Dis-
abled)
   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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaSave (Video Save)                            Stopped
(System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                           Stopped
(System)
   C:\WINNT\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   (Dis-
abled)
   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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process


IRQ and Port Report
-------------------------------------------------------------
-------------
Devices                        Vector Level  Affinity
-------------------------------------------------------------
-------------
MPS 1.4 - APIC platform             8     8 0x000000ff
MPS 1.4 - APIC platform             0     0 0x000000ff
MPS 1.4 - APIC platform             1     1 0x000000ff
MPS 1.4 - APIC platform             2     2 0x000000ff
MPS 1.4 - APIC platform             3     3 0x000000ff
MPS 1.4 - APIC platform             4     4 0x000000ff
MPS 1.4 - APIC platform             5     5 0x000000ff
MPS 1.4 - APIC platform             6     6 0x000000ff
MPS 1.4 - APIC platform             7     7 0x000000ff
MPS 1.4 - APIC platform             8     8 0x000000ff
MPS 1.4 - APIC platform             9     9 0x000000ff
MPS 1.4 - APIC platform            10    10 0x000000ff
MPS 1.4 - APIC platform            11    11 0x000000ff
MPS 1.4 - APIC platform            12    12 0x000000ff
MPS 1.4 - APIC platform            13    13 0x000000ff
MPS 1.4 - APIC platform            14    14 0x000000ff
MPS 1.4 - APIC platform            15    15 0x000000ff
MPS 1.4 - APIC platform            16    16 0x000000ff
MPS 1.4 - APIC platform            17    17 0x000000ff
MPS 1.4 - APIC platform            18    18 0x000000ff
MPS 1.4 - APIC platform            19    19 0x000000ff
MPS 1.4 - APIC platform            20    20 0x000000ff
MPS 1.4 - APIC platform            21    21 0x000000ff
MPS 1.4 - APIC platform            22    22 0x000000ff
MPS 1.4 - APIC platform            23    23 0x000000ff
MPS 1.4 - APIC platform            24    24 0x000000ff
MPS 1.4 - APIC platform            25    25 0x000000ff
MPS 1.4 - APIC platform            26    26 0x000000ff
MPS 1.4 - APIC platform            27    27 0x000000ff
MPS 1.4 - APIC platform            28    28 0x000000ff
MPS 1.4 - APIC platform            29    29 0x000000ff
MPS 1.4 - APIC platform            30    30 0x000000ff
MPS 1.4 - APIC platform            31    31 0x000000ff
MPS 1.4 - APIC platform            32    32 0x000000ff
MPS 1.4 - APIC platform            33    33 0x000000ff
MPS 1.4 - APIC platform            34    34 0x000000ff
MPS 1.4 - APIC platform            35    35 0x000000ff
MPS 1.4 - APIC platform            36    36 0x000000ff
MPS 1.4 - APIC platform            37    37 0x000000ff
MPS 1.4 - APIC platform            38    38 0x000000ff
MPS 1.4 - APIC platform            39    39 0x000000ff
MPS 1.4 - APIC platform            40    40 0x000000ff
MPS 1.4 - APIC platform            41    41 0x000000ff
MPS 1.4 - APIC platform            42    42 0x000000ff
MPS 1.4 - APIC platform            43    43 0x000000ff
MPS 1.4 - APIC platform            44    44 0x000000ff
MPS 1.4 - APIC platform            45    45 0x000000ff
MPS 1.4 - APIC platform            46    46 0x000000ff
MPS 1.4 - APIC platform            47    47 0x000000ff
MPS 1.4 - APIC platform            61    61 0x000000ff
MPS 1.4 - APIC platform            65    65 0x000000ff
MPS 1.4 - APIC platform            80    80 0x000000ff
MPS 1.4 - APIC platform           193   193 0x000000ff
MPS 1.4 - APIC platform           225   225 0x000000ff
MPS 1.4 - APIC platform           253   253 0x000000ff
MPS 1.4 - APIC platform           254   254 0x000000ff
MPS 1.4 - APIC platform           255   255 0x000000ff
i8042prt                            1     1 0xffffffff
i8042prt                           12    12 0xffffffff
Serial                              4     4 0x00000000
Serial                              3     3 0x00000000
Floppy                              6     6 0x00000000
HPTX                               24    24 0x00000000
dac960nt                           16    16 0x00000000
dac960nt                           20    20 0x00000000
dac960nt                           24    24 0x00000000
dac960nt                           12    12 0x00000000
```

```
dac960nt                          16    16 0x00000000
dac960nt                          20    20 0x00000000
symc8XX                           12    12 0x00000000
symc8XX                           16    16 0x00000000
symc8XX                           20    20 0x00000000
symc8XX                           24    24 0x00000000
symc8XX                           12    12 0x00000000
symc8XX                           16    16 0x00000000
symc8XX                           20    20 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
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
HPTX                            0x0000d4e0  0x0000000014
symc8XX                         0x0000f800  0x0000000100
symc8XX                         0x0000f400  0x0000000100
symc8XX                         0x0000f000  0x0000000100
symc8XX                         0x0000e800  0x0000000100
symc8XX                         0x0000e400  0x0000000100
symc8XX                         0x0000e000  0x0000000100
symc8XX                         0x0000d800  0x0000000100
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          0xfec01000  0x00000400
MPS 1.4 - APIC platform          0xfee00000  0x00000400
HPTX                             0xfcafe000  0x00000014
dac960nt                         0xfeafe000  0x00002000
dac960nt                         0xfeafc000  0x00002000
dac960nt                         0xfeafa000  0x00002000
dac960nt                         0xfcafc000  0x00002000
dac960nt                         0xfcafa000  0x00002000
dac960nt                         0xfcaf8000  0x00002000
symc8XX                          0xfceff800  0x00000100
symc8XX                          0xfcefe000  0x00001000
symc8XX                          0xfceff400  0x00000100
symc8XX                          0xfcefd000  0x00001000
symc8XX                          0xfceff000  0x00000100
symc8XX                          0xfcefc000  0x00001000
symc8XX                          0xfcefac00  0x00000100
symc8XX                          0xfcefb000  0x00001000
symc8XX                          0xfcdff800  0x00000100
symc8XX                          0xfcdfe000  0x00001000
symc8XX                          0xfcdff400  0x00000100
symc8XX                          0xfcdfd000  0x00001000
symc8XX                          0xfcdff000  0x00000100
symc8XX                          0xfcdfc000  0x00001000
cirrus                           0x000a0000  0x00020000
cirrus                           0xfd000000  0x01000000


Environment Report
------------------------------------------------------------
-------------



System Environment Variables
   ComSpec=C:\WINNT\system32\cmd.exe
   Os2LibPath=C:\WINNT\system32\os2\dll;

Path=C:\mksnt\mksnt;C:\WINNT\system32;C:\WINNT;;C:\NTRES-
KIT;C:\NTRESKIT\Perl;;C:\MSSQL\BINN
   windir=C:\WINNT
   OS=Windows_NT
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_LEVEL=6
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 1 Stepping 9,
GenuineIntel
   PROCESSOR_REVISION=0109
   NUMBER_OF_PROCESSORS=8
   ROOTDIR=C:/mksnt
   SHELL=C:/mksnt/mksnt/sh.exe
   HOME=C:/
   TMPDIR=C:/TEMP
   NTRESKIT=C:\NTRESKIT



Environment Variables for Current User
   TEMP=C:\TEMP
   TMP=C:\TEMP
   MSDevDir=C:\Program Files\DevStudio\SharedIDE
   path=c:\program files\devstudio\share-
dide\bin\ide;c:\program files\devstudio\share-
dide\bin;c:\program files\devstudio\vc\bin
   lib=c:\program files\devstudio\vc\lib;c:\program
files\devstudio\vc\mfc\lib;c:\program files\devstu-
dio\vc\lib;c:\program files\devstudio\vc\mfc\lib;%lib%
   include=c:\program files\devstudio\vc\include;c:\pro-
gram files\devstudio\vc\atl\include;c:\program files\dev-
studio\vc\mfc\include;c:\program
files\devstudio\vc\include;c:\program files\devstu-
dio\vc\atl\include;c:\program files\devstu-
dio\vc\mfc\include;%include%



Network Report
------------------------------------------------------------
-------------
Your Access Level: Admin & Local
Workgroup or Domain: WORKGROUP
Network Version: 4.0
LanRoot: WORKGROUP
Logged On Users: 5
Current User (1): Administrator
  Logon Domain: PRF_SUT6
  Logon Server: PRF_SUT6
Current User (2): tpcc
  Logon Domain: PRF_SUT6
```

```
  Logon Server: PRF_SUT6
Current User (3): tpcc
  Logon Domain: PRF_SUT6
  Logon Server: PRF_SUT6
Current User (4): tpcc
  Logon Domain: PRF_SUT6
  Logon Server: PRF_SUT6
Current User (5): tpcc
  Logon Domain: PRF_SUT6
  Logon Server: PRF_SUT6
Transport: NetBT_HPTX1, 08-00-09-DB-88-23, VC's: 1, 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: 511,199
SMB's Received: 1,810
Paged Read Bytes Requested: 397,312
Non Paged Read Bytes Requested: 44,544
Cache Read Bytes Requested: 21,306
Network Read Bytes Requested: 340,115
Bytes Transmitted: 2,862,900
SMB's Transmitted: 1,810
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 38,139,743
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 38,055,422
Initally Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 34
Random Read Operations: 2
Read SMB's: 24
Large Read SMB's: 11
Small Read SMB's: 8
Write Operations: 1,552
Random Write Operations: 0
Write SMB's: 637
Large Write SMB's: 623
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 26
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 10
Server Disconnects: 8
Hung Sessions: 0
Use Count: 3
Failed Use Count: 0
Current Commands: 0
Server File Opens: 0
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 0
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: 269
Server Bytes Received: 489
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

Microsoft SQL Server Version 6.5 Startup Parameters

```
c:\mssql\binn\sqlservr -c -x -t1081 -t3502 -t812 -T1140 -
Cp4500 -Cd144000
where
-c          start SQL Server
            independently of the
            Windows NT 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
-t3052      prints a message to the log at
            the start and end of each
            checkpoint
-t812       omits sorting for write page
            ordering during checkpoints
-T1140      Optimizes free space allocation
-Cp4500     specifies number of procedure
            cache buffers to allocate
-Cd144000   specifies number of data
            buffers to allocate
```

Cache Column of Sysobjects Table

Before the Benchmark was run the following script was executed to improve the cache performance of tables.

```
use tpcc
go
update sysobjects set cache=5 from sysobjects where
name="customer"
go
update sysobjects set cache=2 from sysobjects where
name="stock"
go
update sysobjects set cache=2 from sysobjects where
name="orders"
go
update sysobjects set cache=2 from sysobjects where
name="order_line"
go
update sysobjects set cache=2 from sysobjects where
name="new_order"
go
```

Microsoft SQL Server6.5 Stack Size

The default stack size for Microsoft SQLServer6.5 Enterprise Edition 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: 16384 sqlservr.exe
```

Microsoft SQL Server Version 6.5 Configuration Parameters

```
1> 2> sp_configure
 name                                minimum     maximum
    config_value
run_value
 ----------------------------------- ----------- --------
    --- ------------
-----------
 affinity mask                         0  2147483647
    255
         255
 allow updates                          0         1
    1
           1
 backup buffer size                     1        32
5
           5
 backup threads                         0        32
0
           0
 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
4000
        4000
 hash buckets                        4999    1000000
1000000
    1000003
 language in cache                      3       100
3
           3
 LE threshold maximum                   2    500000
300
         300
 LE threshold minimum                   2    500000
20
          20
 LE threshold percent                   1       100
0
           0
 locks                               5000  2147483647
7000
        7000
 LogLRU buffers                         0  2147483647
5250
        5250
 logwrite sleep (ms)                   -1       500
-1
          -1
 max async IO                           1      1024
16
          16
 max lazywrite IO                       1      1024
100
         100
 max text repl size                     0  2147483647
65536
       65536
 max worker threads                    10      1024
250
         250
 media retention                        0       365
0
           0
 memory                              2800   1048576
800000
      800000
 nested triggers                        0         1
1
           1
 network packet size                  512     32767
4096
        4096
 open databases                         5     32767
10
          10
 open objects                         100  2147483647
500
         500
 priority boost                         0         1
0
           0
 procedure cache                        1        99
30
          30
 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
3
           3
 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
280
        280
 user options                             0      4095
0
          0

(1 row affected)
```

Disk Array Configuration Parameters

Internal DACs

```
************************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility          *
     *                              Version
4.76                            *

************************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #1    Firmware ver-
sion 4.03


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:8]  [0:9]
[0:10]  [0:11]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:8]  [1:9]
[1:10]  [1:11]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]
[2:10]  [2:11]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

    Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write
Policy
     =========   =========   ==========   =========
===========
         0        208272  MB       0        208272  MB
Write Thru



************************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility          *
     *                              Version
4.76                            *

************************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #2    Firmware ver-
sion 4.03


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:8]  [0:9]
[0:10]  [0:11]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:8]  [1:9]
[1:10]  [1:11]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]
[2:10]  [2:11]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

    Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write
Policy
     =========   =========   ==========   =========
===========
         0        97608  MB       0        97608  MB   Write
Thru



************************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility          *
     *                              Version
4.76                            *

************************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #3    Firmware ver-
sion 4.03


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:8]  [0:9]
[0:10]  [0:11]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:8]  [1:9]
```

```
[1:10]  [1:11]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]
[2:10]  [2:11]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

    Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write
Policy
     =========   =========   ==========   =========
============
         0       97608  MB       0       97608  MB   Write
Thru


*********************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility         *
     *                            Version
4.76                           *

*********************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #4    Firmware ver-
sion 4.03


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:8]  [0:9]
[0:10]  [0:11]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:8]  [1:9]
[1:10]  [1:11]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]
[2:10]  [2:11]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

    Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write
Policy
     =========   =========   ==========   =========
============
          0        208272  MB       0        208272  MB
Write Thru



*********************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility         *
     *                            Version
4.76                           *

*********************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #5    Firmware ver-
sion 4.03


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:8]  [0:9]
[0:10]  [0:11]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:8]  [1:9]
[1:10]  [1:11]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]
[2:10]  [2:11]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

    Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write
Policy
     =========   =========   ==========   =========
============
          0        208272  MB       0        208272  MB
Write Thru



*********************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility         *
     *                            Version
4.76                           *

*********************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #6    Firmware ver-
sion 4.03


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:8]  [0:9]
[0:10]  [0:11]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:8]  [1:9]
[1:10]  [1:11]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]
[2:10]  [2:11]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

    Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write
Policy
     =========   =========   ==========   =========
============
          0        208272  MB       0        208272  MB
Write Thru
```

External DACs

```
*********************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility         *
     *                            Version
7.11                           *
```

```
************************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     5 Channel - 16 Target  DAC960SX  Firmware version
3.24-00

     Daughter Card


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 1

     Pack A : [2:0]  [3:0]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1
     LUN Affinity = Disabled

     # Phy. Size  Raid  Eff. Size  Cache       Packs  Sta-
tus
     = =========  ====  =========  ==========  =====
======
     0  86778 MB    1    43389 MB  Write Thru  A...   Onln

     SYSTEM DRIVE AFFINITY INFORMATION :
     ===================================

     All Affinity (No LUN mapping)

     PHYSICAL DRIVE INFORMATION :
     ============================

     Chn  Tgt  Present  Type  Status   Size(MB)  Size(RAW)
     ===  ===  =======  ====  =======  ========  =========
      0    0   Absent    00   DEAD           0    00000000
      0    1   Absent    00   DEAD           0    00000000
      0    2   Absent    00   DEAD           0    00000000
      0    3   Absent    00   DEAD           0    00000000
      0    4   Absent    00   DEAD           0    00000000
      0    5   Absent    00   DEAD           0    00000000
      0    6   Absent    00   DEAD           0    00000000
      0    8   Absent    00   DEAD           0    00000000
      0    9   Absent    00   DEAD           0    00000000
      0    a   Absent    00   DEAD           0    00000000
      0    b   Absent    00   DEAD           0    00000000
      0    c   Absent    00   DEAD           0    00000000
      0    d   Absent    00   DEAD           0    00000000
      0    e   Absent    00   DEAD           0    00000000
      0    f   Absent    00   DEAD           0    00000000
      1    0   Absent    00   DEAD           0    00000000
      1    1   Absent    00   DEAD           0    00000000
      1    2   Absent    00   DEAD           0    00000000
      1    3   Absent    00   DEAD           0    00000000
      1    4   Absent    00   DEAD           0    00000000
      1    5   Absent    00   DEAD           0    00000000
      1    6   Absent    00   DEAD           0    00000000
      1    8   Absent    00   DEAD           0    00000000
      1    9   Absent    00   DEAD           0    00000000
      1    a   Absent    00   DEAD           0    00000000
      1    b   Absent    00   DEAD           0    00000000
      1    c   Absent    00   DEAD           0    00000000
      1    d   Absent    00   DEAD           0    00000000
      1    e   Absent    00   DEAD           0    00000000
      1    f   Absent    00   DEAD           0    00000000
      2    0   Present   f1   ONLINE     43389    054be800
      2    1   Absent    00   DEAD           0    00000000
      2    2   Absent    00   DEAD           0    00000000
      2    3   Absent    00   DEAD           0    00000000
      2    4   Absent    00   DEAD           0    00000000
      2    5   Absent    00   DEAD           0    00000000
      2    6   Absent    00   DEAD           0    00000000
      2    8   Absent    00   DEAD           0    00000000
      2    9   Absent    00   DEAD           0    00000000
      2    a   Absent    00   DEAD           0    00000000
      2    b   Absent    00   DEAD           0    00000000
      2    c   Absent    00   DEAD           0    00000000
      2    d   Absent    00   DEAD           0    00000000
      2    e   Absent    00   DEAD           0    00000000
      2    f   Absent    00   DEAD           0    00000000
      3    0   Present   f1   ONLINE     43389    054be800
      3    1   Absent    00   DEAD           0    00000000
      3    2   Absent    00   DEAD           0    00000000
      3    3   Absent    00   DEAD           0    00000000
      3    4   Absent    00   DEAD           0    00000000
      3    5   Absent    00   DEAD           0    00000000
      3    6   Absent    00   DEAD           0    00000000
      3    8   Absent    00   DEAD           0    00000000
      3    9   Absent    00   DEAD           0    00000000
      3    a   Absent    00   DEAD           0    00000000
      3    b   Absent    00   DEAD           0    00000000
      3    c   Absent    00   DEAD           0    00000000
      3    d   Absent    00   DEAD           0    00000000
      3    e   Absent    00   DEAD           0    00000000
      3    f   Absent    00   DEAD           0    00000000
      4    0   Absent    00   DEAD           0    00000000
      4    1   Absent    00   DEAD           0    00000000
      4    2   Absent    00   DEAD           0    00000000
      4    3   Absent    00   DEAD           0    00000000
      4    4   Absent    00   DEAD           0    00000000
      4    5   Absent    00   DEAD           0    00000000
      4    6   Absent    00   DEAD           0    00000000
      4    8   Absent    00   DEAD           0    00000000
      4    9   Absent    00   DEAD           0    00000000
      4    a   Absent    00   DEAD           0    00000000
      4    b   Absent    00   DEAD           0    00000000
      4    c   Absent    00   DEAD           0    00000000
      4    d   Absent    00   DEAD           0    00000000
      4    e   Absent    00   DEAD           0    00000000
      4    f   Absent    00   DEAD           0    00000000

     CONTROLLER PARAMETERS :
     =======================

        Hardware

    Automatic Rebuild Management     Enabled
    Operational Fault Management     Enabled
    Disconnect on First Command     Disabled

        Physical

    Default rebuild rate                 50
    Controller read ahead             Enabled
    Super read ahead                 Disabled
    True Verification of Data        Disabled
    Stripe Size(K bytes)                 16
    Installation Abort               Disabled
    Reassign Restricted to 1 blk      Enabled
    Write Through Verify              Enabled
    RAID 5 Algorithm              Right Asym

        Disk Side Parameters

    Per Channel:            0     1     2     3     4
     Data Transfer Rate - 10MHz 10MHz 10MHz 10MHz 10MHz
     Command tagging    - Enabl Enabl Enabl Enabl Enabl
     Data bus width     - 16bit 16bit 16bit 16bit 16bit
    Per Device:
     Elevator   -  Disab
     Coalescing -  Disab
     Que Limit  -    0
    Global:
     Max IOPs       -   0
     Spin Up Option - Automatic  /  2 /  6 /  0

        Host Side
```

```
    Disable Wide Operation          Disabled
    Vendor Unique TUR               Disabled
    Disable CC for Invalid LUN       Enabled
    No Pause on ctrlr not ready     Disabled
    On Queue Full give Busy         Disabled

        Serial port 0



        Active - Active

    Conservative Cache              Disabled
    Auto Failback                   Disabled
    Force Simplex                    Enabled
    Host Bus Reset Delay                   0
    Ctrlr Pres/Flt Signals          Disabled
    Ctrlr Pres/Flt Select                  A
    Simplex no RSTCOM               Disabled

        Fibre

    PCI Latency Control                Short
    Frame Control                       Long
    Smart Large Transfers            Enabled
    Port 0                          Disabled
    Port 0 ID                              0
    Port 1                          Disabled
    Port 1 ID                              0




************************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility         *
     *                            Version
7.11                           *

************************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     5 Channel - 16 Target  DAC960SX  Firmware version
3.24-00

     Daughter Card


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 1

     Pack A : [2:0]   [2:1]   [2:2]   [3:0]   [3:1]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1
     LUN Affinity = Disabled

     # Phy. Size  Raid  Eff. Size  Cache       Packs  Sta-
tus
     = =========  ====  =========  ==========  =====
======
     0  43390 MB    0    43390 MB  Write Back  A...   Onln

     SYSTEM DRIVE AFFINITY INFORMATION :
     ===================================

     All Affinity (No LUN mapping)

     PHYSICAL DRIVE INFORMATION :
     ============================

Chn  Tgt  Present  Type  Status   Size(MB)  Size(RAW)
===  ===  =======  ====  =======  ========  =========
  0    0  Absent    00   DEAD            0   00000000
  0    1  Absent    00   DEAD            0   00000000
  0    2  Absent    00   DEAD            0   00000000
  0    3  Absent    00   DEAD            0   00000000
  0    4  Absent    00   DEAD            0   00000000
  0    5  Absent    00   DEAD            0   00000000
  0    6  Absent    00   DEAD            0   00000000
  0    8  Absent    00   DEAD            0   00000000
  0    9  Absent    00   DEAD            0   00000000
  0    a  Absent    00   DEAD            0   00000000
  0    b  Absent    00   DEAD            0   00000000
  0    c  Absent    00   DEAD            0   00000000
  0    d  Absent    00   DEAD            0   00000000
  0    e  Absent    00   DEAD            0   00000000
  0    f  Absent    00   DEAD            0   00000000
  1    0  Absent    00   DEAD            0   00000000
  1    1  Absent    00   DEAD            0   00000000
  1    2  Absent    00   DEAD            0   00000000
  1    3  Absent    00   DEAD            0   00000000
  1    4  Absent    00   DEAD            0   00000000
  1    5  Absent    00   DEAD            0   00000000
  1    6  Absent    00   DEAD            0   00000000
  1    8  Absent    00   DEAD            0   00000000
  1    9  Absent    00   DEAD            0   00000000
  1    a  Absent    00   DEAD            0   00000000
  1    b  Absent    00   DEAD            0   00000000
  1    c  Absent    00   DEAD            0   00000000
  1    d  Absent    00   DEAD            0   00000000
  1    e  Absent    00   DEAD            0   00000000
  1    f  Absent    00   DEAD            0   00000000
  2    0  Present   f1   ONLINE     8678    010f3000
  2    1  Present   f1   ONLINE     8678    010f3000
  2    2  Present   f1   ONLINE     8678    010f3000
  2    3  Absent    00   DEAD            0   00000000
  2    4  Absent    00   DEAD            0   00000000
  2    5  Absent    00   DEAD            0   00000000
  2    6  Absent    00   DEAD            0   00000000
  2    8  Absent    00   DEAD            0   00000000
  2    9  Absent    00   DEAD            0   00000000
  2    a  Absent    00   DEAD            0   00000000
  2    b  Absent    00   DEAD            0   00000000
  2    c  Absent    00   DEAD            0   00000000
  2    d  Absent    00   DEAD            0   00000000
  2    e  Absent    00   DEAD            0   00000000
  2    f  Absent    00   DEAD            0   00000000
  3    0  Present   f1   ONLINE     8678    010f3000
  3    1  Present   f1   ONLINE     8678    010f3000
  3    2  Present   f1   STANDBY    8678    010f3000
  3    3  Absent    00   DEAD            0   00000000
  3    4  Absent    00   DEAD            0   00000000
  3    5  Absent    00   DEAD            0   00000000
  3    6  Absent    00   DEAD            0   00000000
  3    8  Absent    00   DEAD            0   00000000
  3    9  Absent    00   DEAD            0   00000000
  3    a  Absent    00   DEAD            0   00000000
  3    b  Absent    00   DEAD            0   00000000
  3    c  Absent    00   DEAD            0   00000000
  3    d  Absent    00   DEAD            0   00000000
  3    e  Absent    00   DEAD            0   00000000
  3    f  Absent    00   DEAD            0   00000000
  4    0  Absent    00   DEAD            0   00000000
  4    1  Absent    00   DEAD            0   00000000
  4    2  Absent    00   DEAD            0   00000000
  4    3  Absent    00   DEAD            0   00000000
  4    4  Absent    00   DEAD            0   00000000
  4    5  Absent    00   DEAD            0   00000000
  4    6  Absent    00   DEAD            0   00000000
  4    8  Absent    00   DEAD            0   00000000
  4    9  Absent    00   DEAD            0   00000000
  4    a  Absent    00   DEAD            0   00000000
  4    b  Absent    00   DEAD            0   00000000
  4    c  Absent    00   DEAD            0   00000000
  4    d  Absent    00   DEAD            0   00000000
```

```
     4    e   Absent    00   DEAD          0    00000000
     4    f   Absent    00   DEAD          0    00000000

    CONTROLLER PARAMETERS :
    ======================

       Hardware

    Automatic Rebuild Management     Enabled
    Operational Fault Management     Enabled
    Disconnect on First Command     Disabled

       Physical

    Default rebuild rate                50
    Controller read ahead             Enabled
    Super read ahead                 Disabled
    True Verification of Data        Disabled
    Stripe Size(K bytes)                16
    Installation Abort               Disabled
    Reassign Restricted to 1 blk      Enabled
    Write Through Verify             Disabled
    RAID 5 Algorithm               Right Asym

       Disk Side Parameters

    Per Channel:            0     1     2     3     4
     Data Transfer Rate - 10MHz 10MHz 10MHz 10MHz 10MHz
     Command tagging    - Enabl Enabl Enabl Enabl Enabl
     Data bus width     - 16bit 16bit 16bit 16bit 16bit
    Per Device:
     Elevator   -  Disab
     Coalescing -  Disab
     Que Limit  -    8
    Global:
     Max IOPs       -   0
     Spin Up Option - Automatic  /  2 /  6 /  0

       Host Side

    Disable Wide Operation           Disabled
    Vendor Unique TUR                Disabled
    Disable CC for Invalid LUN        Enabled
    No Pause on ctrlr not ready      Disabled
    On Queue Full give Busy          Disabled

       Serial port 0



       Active - Active

    Conservative Cache               Disabled
    Auto Failback                    Disabled
    Force Simplex                     Enabled
    Host Bus Reset Delay                    0
    Ctrlr Pres/Flt Signals           Disabled
    Ctrlr Pres/Flt Select                   A
    Simplex no RSTCOM                Disabled

       Fibre

    PCI Latency Control                 Short
    Frame Control                        Long
    Smart Large Transfers             Enabled
    Port 0                           Disabled
    Port 0 ID                               0
    Port 1                           Disabled
    Port 1 ID                               0



************************************************************
*************
     *        MYLEX Disk Array Controller - Configuration
Utility         *
     *                              Version
7.11                             *

************************************************************
*************

     CONFIGURATION INFORMATION OF :
     ==============================

     5 Channel - 16 Target  DAC960SX  Firmware version
3.24-00

     Daughter Card


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 1

     Pack A : [2:0]  [2:1]  [2:2]  [3:0]  [3:1]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1
     LUN Affinity = Disabled

     # Phy. Size  Raid  Eff. Size  Cache       Packs  Sta-
tus
     = =========  ====  =========  ==========  =====
======
     0  43390 MB    0    43390 MB  Write Back  A...   Onln

     SYSTEM DRIVE AFFINITY INFORMATION :
     ===================================

     All Affinity (No LUN mapping)

     PHYSICAL DRIVE INFORMATION :
     ============================

     Chn  Tgt  Present  Type  Status   Size(MB)  Size(RAW)
     ===  ===  =======  ====  =======  ========  =========
      0    0   Absent    00   DEAD           0    00000000
      0    1   Absent    00   DEAD           0    00000000
      0    2   Absent    00   DEAD           0    00000000
      0    3   Absent    00   DEAD           0    00000000
      0    4   Absent    00   DEAD           0    00000000
      0    5   Absent    00   DEAD           0    00000000
      0    6   Absent    00   DEAD           0    00000000
      0    8   Absent    00   DEAD           0    00000000
      0    9   Absent    00   DEAD           0    00000000
      0    a   Absent    00   DEAD           0    00000000
      0    b   Absent    00   DEAD           0    00000000
      0    c   Absent    00   DEAD           0    00000000
      0    d   Absent    00   DEAD           0    00000000
      0    e   Absent    00   DEAD           0    00000000
      0    f   Absent    00   DEAD           0    00000000
      1    0   Absent    00   DEAD           0    00000000
      1    1   Absent    00   DEAD           0    00000000
      1    2   Absent    00   DEAD           0    00000000
      1    3   Absent    00   DEAD           0    00000000
      1    4   Absent    00   DEAD           0    00000000
      1    5   Absent    00   DEAD           0    00000000
      1    6   Absent    00   DEAD           0    00000000
      1    8   Absent    00   DEAD           0    00000000
      1    9   Absent    00   DEAD           0    00000000
      1    a   Absent    00   DEAD           0    00000000
      1    b   Absent    00   DEAD           0    00000000
      1    c   Absent    00   DEAD           0    00000000
      1    d   Absent    00   DEAD           0    00000000
      1    e   Absent    00   DEAD           0    00000000
      1    f   Absent    00   DEAD           0    00000000
      2    0   Present   f1   ONLINE      8678    010f3000
      2    1   Present   f1   ONLINE      8678    010f3000
      2    2   Present   f1   ONLINE      8678    010f3000
```

```
  2    3   Absent    00   DEAD          0    00000000
  2    4   Absent    00   DEAD          0    00000000
  2    5   Absent    00   DEAD          0    00000000
  2    6   Absent    00   DEAD          0    00000000
  2    8   Absent    00   DEAD          0    00000000
  2    9   Absent    00   DEAD          0    00000000
  2    a   Absent    00   DEAD          0    00000000
  2    b   Absent    00   DEAD          0    00000000
  2    c   Absent    00   DEAD          0    00000000
  2    d   Absent    00   DEAD          0    00000000
  2    e   Absent    00   DEAD          0    00000000
  2    f   Absent    00   DEAD          0    00000000
  3    0   Present   f1   ONLINE     8678    010f3000
  3    1   Present   f1   ONLINE     8678    010f3000
  3    2   Present   f1   STANDBY    8678    010f3000
  3    3   Absent    00   DEAD          0    00000000
  3    4   Absent    00   DEAD          0    00000000
  3    5   Absent    00   DEAD          0    00000000
  3    6   Absent    00   DEAD          0    00000000
  3    8   Absent    00   DEAD          0    00000000
  3    9   Absent    00   DEAD          0    00000000
  3    a   Absent    00   DEAD          0    00000000
  3    b   Absent    00   DEAD          0    00000000
  3    c   Absent    00   DEAD          0    00000000
  3    d   Absent    00   DEAD          0    00000000
  3    e   Absent    00   DEAD          0    00000000
  3    f   Absent    00   DEAD          0    00000000
  4    0   Absent    00   DEAD          0    00000000
  4    1   Absent    00   DEAD          0    00000000
  4    2   Absent    00   DEAD          0    00000000
  4    3   Absent    00   DEAD          0    00000000
  4    4   Absent    00   DEAD          0    00000000
  4    5   Absent    00   DEAD          0    00000000
  4    6   Absent    00   DEAD          0    00000000
  4    8   Absent    00   DEAD          0    00000000
  4    9   Absent    00   DEAD          0    00000000
  4    a   Absent    00   DEAD          0    00000000
  4    b   Absent    00   DEAD          0    00000000
  4    c   Absent    00   DEAD          0    00000000
  4    d   Absent    00   DEAD          0    00000000
  4    e   Absent    00   DEAD          0    00000000
  4    f   Absent    00   DEAD          0    00000000

 CONTROLLER PARAMETERS :
 =======================

    Hardware

Automatic Rebuild Management     Enabled
Operational Fault Management     Enabled
Disconnect on First Command     Disabled

    Physical

Default rebuild rate                 50
Controller read ahead             Enabled
Super read ahead                 Disabled
True Verification of Data        Disabled
Stripe Size(K bytes)                 16
Installation Abort               Disabled
Reassign Restricted to 1 blk      Enabled
Write Through Verify              Enabled
RAID 5 Algorithm              Right Asym

    Disk Side Parameters

Per Channel:            0     1     2     3     4
 Data Transfer Rate - 10MHz 10MHz 10MHz 10MHz 10MHz
 Command tagging    - Enabl Enabl Enabl Enabl Enabl
 Data bus width     - 16bit 16bit 16bit 16bit 16bit
Per Device:
 Elevator   -  Disab
 Coalescing -  Disab
 Que Limit  -    0
Global:
 Max IOPs       -   0
 Spin Up Option - Automatic  /  2 /  6 /  0

    Host Side

    Disable Wide Operation          Disabled
    Vendor Unique TUR               Disabled
    Disable CC for Invalid LUN       Enabled
    No Pause on ctrlr not ready     Disabled
    On Queue Full give Busy         Disabled

    Serial port 0



    Active - Active

    Conservative Cache              Disabled
    Auto Failback                   Disabled
    Force Simplex                    Enabled
    Host Bus Reset Delay                   0
    Ctrlr Pres/Flt Signals          Disabled
    Ctrlr Pres/Flt Select                 A
    Simplex no RSTCOM               Disabled

    Fibre

    PCI Latency Control                Short
    Frame Control                       Long
    Smart Large Transfers            Enabled
    Port 0                          Disabled
    Port 0 ID                              0
    Port 1                          Disabled
    Port 1 ID                              0
```

HP NetServer LCII Configurations - Clients

```
Microsoft Diagnostics Report For \\UWLC1
------------------------------------------------------------
-------------

OS Version Report
------------------------------------------------------------
-------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiproces-
sor Free
Registered Owner: Hewlett Packard, Network Server Division
Product Number: 50370-111-1111111-91655
------------------------------------------------------------
-------------

System Report
------------------------------------------------------------
-------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 10/13/97
BIOS Version: <unavailable>

Processor list:
   0:  x86 Family 6 Model 3 Stepping 4 GenuineIntel ~266
Mhz
   1:  x86 Family 6 Model 3 Stepping 3 GenuineIntel ~266
Mhz
------------------------------------------------------------
-------------

Video Display Report
------------------------------------------------------------
-------------
BIOS Date: 05/21/97
BIOS Version: CL-GD5446 PCI VGA BIOS Version 1.33
```

```
Adapter:
   Setting: 800 x 600 x 16
            Hardware Default Refresh
   Type: vga compatible display adapter
   String: <unavailable>
   Memory:
   Chip Type: <unavailable>
   DAC Type: <unavailable>
Driver:
   Vendor: Microsoft Corporation
   File(s): vga.sys, vga.dll
   Version: 4.00, 4.0.0


Drives Report
-------------------------------------------------------------
-------------
C:\  (Local - NTFS) TPCC_CLIENT Total: 0KB, Free: 0KB
   Serial Number: 32CA - A0EF
   Bytes per cluster: 512
   Sectors per cluster: 1
   Filename length: 255
W:  (Remote - NTFS) \\prf_sut6\c$  Total: 7,831,684KB,
Free: 2,777,324KB
   Serial Number: 2CDD - 9F3A
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
X:  (Remote - NTFS) \\nrddata\f$  Total: 8,886,252KB,
Free: 966,916KB
   Serial Number: 640E - B515
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255

Memory Report
-------------------------------------------------------------
-------------
Handles: 20,245
Threads: 144
Processes: 17

Physical Memory (K)
   Total: 523,696
   Available: 450,004
   File Cache: 12,756

Kernel Memory (K)
   Total: 22,944
   Paged: 7,828
   Nonpaged: 15,116

Commit Charge (K)
   Total: 51,028
   Limit: 905,000
   Peak: 51,296

Pagefile Space (K)
   Total: 409,600
   Total in use: 2,968
   Peak: 3,064

   C:\pagefile.sys
      Total: 409,600
      Total in use: 2,968
      Peak: 3,064



Services Report
-------------------------------------------------------------
-------------
Alerter                                          Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
Ataman TCP Remote Logon Services                 Running
(Automatic)
   C:\atrls2\ATRLS.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Computer Browser                                 Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
      LmHosts
ClipBook Server                                  Stopped
(Manual)
   C:\WINNT\system32\clipsrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      NetDDE
DHCP Client (TDI)                                Stopped
(Disabled)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Tcpip
      Afd
      NetBT
EventLog (Event log)                             Running
(Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Gopher Publishing Service                        Running
(Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
Server                                           Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
Workstation (NetworkProvider)                    Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
License Logging Service                          Running
(Automatic)
   C:\WINNT\System32\llssrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TCP/IP NetBIOS Helper                            Running
(Automatic)
   C:\WINNT\System32\services.exe
```

```
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      NetworkProvider
Messenger                                   Running
(Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      NetBios
Monitor Service                             Stopped
(Manual)
   C:\WINNT\system32\datalog.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
FTP Publishing Service                      Running
(Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
MSSQLServer                            Stopped   (Dis-
abled)
   c:\MSSQL\BINN\SQLSERVR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
Network DDE (NetDDEGroup)                   Stopped
(Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      NetDDEDSDM
Network DDE DSDM                            Stopped
(Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Net Logon (RemoteValidation)                Stopped
(Manual)
   C:\WINNT\System32\lsass.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LmHosts
NT LM Security Support Provider             Running
(Manual)
   C:\WINNT\System32\SERVICES.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Plug and Play (PlugPlay)                    Running
(Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Directory Replicator                        Stopped
(Manual)
   C:\WINNT\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)
   C:\WINNT\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)
   C:\WINNT\system32\RpcSs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Schedule                               Stopped   (Man-
ual)
   C:\WINNT\System32\AtSvc.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Spooler (SpoolerGroup)                      Running
(Automatic)
   C:\WINNT\system32\spoolss.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
SQLExecutive                           Stopped   (Man-
ual)
   C:\MSSQL\BINN\SQLEXEC.EXE
   Service Account Name: .\Administrator
   Error Severity: Normal
   Service Flags: Own Process
Telephony Service                           Stopped
(Manual)
   C:\WINNT\system32\tapisrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TUXEDO IPC HELPER                           Running
(Automatic)
   C:\TUXEDO\bin\tuxipc.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TListen (Port: 3050)                        Running
(Automatic)
   C:\TUXEDO\bin\slisten.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
UPS                                    Stopped   (Man-
ual)
   C:\WINNT\System32\ups.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
World Wide Web Publishing Service           Running
(Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP

Drivers Report
-------------------------------------------------------
-------------
Abiosdsk (Primary disk)                     Stopped
(Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
AFD Networking Support Environment (TDI)    Running
```

```
(Automatic)
   C:\WINNT\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)
   C:\WINNT\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
atapi (SCSI miniport)                          Running
(Boot)
   C:\WINNT\System32\DRIVERS\atapi.sys
   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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Beep (Base)                                    Running   (Sys-
tem)
   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)                                 Stopped   (Dis-
abled)
   Error Severity: Ignore
   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
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
et4000 (Video)                                 Stopped   (Dis-
abled)
   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
HP NetServer 10/100TX PCI Ethernet Adapter Driver (NDIS)
Running   (Automatic)
   C:\WINNT\System32\drivers\hptxnt.sys
   Error Severity: Normal
   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   (Sys-
tem)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mga (Video)                                    Stopped   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mga_mil (Video)                                Stopped
(Disabled)
   Error Severity: Normal
   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
Msfs (File system)                             Running
(System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                                  Running   (Man-
ual)
   C:\WINNT\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)               Running
(Manual)
   C:\WINNT\System32\drivers\netbios.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      TDI
WINS Client(TCP/IP) (PNP_TDI)                  Running
(Automatic)
   C:\WINNT\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                                      Stopped   (Man-
ual)
   C:\WINNT\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   (Sys-
tem)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
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
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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Rdr (Network)                                Running   (Man-
ual)
   C:\WINNT\System32\drivers\rdr.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
s3 (Video)                                   Stopped   (Dis-
abled)
   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   (Man-
ual)
   C:\WINNT\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)
   C:\WINNT\System32\drivers\tcpip.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
tga (Video)                                  Stopped   (Dis-
abled)
   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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
vga (Video)                                  Running   (Sys-
tem)
   System32\DRIVERS\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaSave (Video Save)                           Stopped
(System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                          Stopped
(System)
   C:\WINNT\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   (Dis-
abled)
   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   (Dis-
abled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process

IRQ and Port Report
-------------------------------------------------------------
-------------
Devices                       Vector Level  Affinity
-------------------------------------------------------------
```

```
-------------
MPS 1.4 - APIC platform            8     8 0x00000003
MPS 1.4 - APIC platform            0     0 0x00000003
MPS 1.4 - APIC platform            1     1 0x00000003
MPS 1.4 - APIC platform            2     2 0x00000003
MPS 1.4 - APIC platform            3     3 0x00000003
MPS 1.4 - APIC platform            4     4 0x00000003
MPS 1.4 - APIC platform            5     5 0x00000003
MPS 1.4 - APIC platform            6     6 0x00000003
MPS 1.4 - APIC platform            7     7 0x00000003
MPS 1.4 - APIC platform            8     8 0x00000003
MPS 1.4 - APIC platform            9     9 0x00000003
MPS 1.4 - APIC platform           10    10 0x00000003
MPS 1.4 - APIC platform           11    11 0x00000003
MPS 1.4 - APIC platform           12    12 0x00000003
MPS 1.4 - APIC platform           13    13 0x00000003
MPS 1.4 - APIC platform           14    14 0x00000003
MPS 1.4 - APIC platform           15    15 0x00000003
MPS 1.4 - APIC platform           16    16 0x00000003
MPS 1.4 - APIC platform           17    17 0x00000003
MPS 1.4 - APIC platform           18    18 0x00000003
MPS 1.4 - APIC platform           19    19 0x00000003
MPS 1.4 - APIC platform           20    20 0x00000003
MPS 1.4 - APIC platform           21    21 0x00000003
MPS 1.4 - APIC platform           22    22 0x00000003
MPS 1.4 - APIC platform           23    23 0x00000003
MPS 1.4 - APIC platform           24    24 0x00000003
MPS 1.4 - APIC platform           25    25 0x00000003
MPS 1.4 - APIC platform           26    26 0x00000003
MPS 1.4 - APIC platform           27    27 0x00000003
MPS 1.4 - APIC platform           28    28 0x00000003
MPS 1.4 - APIC platform           29    29 0x00000003
MPS 1.4 - APIC platform           30    30 0x00000003
MPS 1.4 - APIC platform           31    31 0x00000003
MPS 1.4 - APIC platform           32    32 0x00000003
MPS 1.4 - APIC platform           33    33 0x00000003
MPS 1.4 - APIC platform           34    34 0x00000003
MPS 1.4 - APIC platform           35    35 0x00000003
MPS 1.4 - APIC platform           36    36 0x00000003
MPS 1.4 - APIC platform           37    37 0x00000003
MPS 1.4 - APIC platform           38    38 0x00000003
MPS 1.4 - APIC platform           39    39 0x00000003
MPS 1.4 - APIC platform           40    40 0x00000003
MPS 1.4 - APIC platform           41    41 0x00000003
MPS 1.4 - APIC platform           42    42 0x00000003
MPS 1.4 - APIC platform           43    43 0x00000003
MPS 1.4 - APIC platform           44    44 0x00000003
MPS 1.4 - APIC platform           45    45 0x00000003
MPS 1.4 - APIC platform           46    46 0x00000003
MPS 1.4 - APIC platform           47    47 0x00000003
MPS 1.4 - APIC platform           61    61 0x00000003
MPS 1.4 - APIC platform           65    65 0x00000003
MPS 1.4 - APIC platform           80    80 0x00000003
MPS 1.4 - APIC platform          193   193 0x00000003
MPS 1.4 - APIC platform          225   225 0x00000003
MPS 1.4 - APIC platform          253   253 0x00000003
MPS 1.4 - APIC platform          254   254 0x00000003
MPS 1.4 - APIC platform          255   255 0x00000003
i8042prt                           1     1 0xffffffff
i8042prt                          12    12 0xffffffff
Serial                             4     4 0x00000000
Serial                             3     3 0x00000000
Floppy                             6     6 0x00000000
HPTX                              32    32 0x00000000
HPTX                              36    36 0x00000000
aic78xx                           40    40 0x00000000
atapi                              0    14 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         0x000000f0  0x0000000010
i8042prt                        0x00000060  0x0000000001
i8042prt                        0x00000064  0x0000000001
Parport                         0x00000378  0x0000000003
Serial                          0x000003f8  0x0000000007
Serial                          0x000002f8  0x0000000007
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
HPTX                            0x0000fce0  0x0000000014
HPTX                            0x0000fcc0  0x0000000014
aic78xx                         0x0000f800  0x0000000100
atapi                           0x000001f0  0x0000000008
atapi                           0x000003f6  0x0000000001
vga                             0x000003b0  0x000000000c
vga                             0x000003c0  0x0000000020
vga                             0x000001ce  0x0000000002



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
HPTX                             0xfecfd000  0x00000014
HPTX                             0xfecfc000  0x00000014
aic78xx                          0xfecff000  0x00001000
vga                              0x000a0000  0x00020000



Environment Report
------------------------------------------------------------
-------------



System Environment Variables
   APPDIR=C:\inetpub\wwwroot
   ComSpec=C:\WINNT\system32\cmd.exe
   HOME=C:/
   NTRESKIT=C:\NTRESKIT
   NUMBER_OF_PROCESSORS=2
   OS=Windows_NT
   Os2LibPath=C:\WINNT\system32\os2\dll;

Path=C:\mksnt;C:\WINNT\system32;C:\WINNT;;C:\MSSQL\BINN;C
:\TUXEDO\bin;C:\NTRESKIT
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 3 Stepping 4,
GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=0303
   ROOTDIR=C:/
   SHELL=C:/mksnt/sh.exe
   TMCONTEXTS=1
   TMPDIR=C:/TEMP
   TUXCONFIG=C:\inetpub\wwwroot\tuxconfig
   windir=C:\WINNT
   TUXDIR=C:\TUXEDO
```

```
Environment Variables for Current User

Path=C:\mksnt;C:\WINNT\system32;C:\WINNT;C:\MSSQL\BINN;C:
\TUXEDO\bin;C:\NTRESKIT
   TEMP=C:\TEMP
   TMP=C:\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: UWLC1
  Logon Server: UWLC1

Transport: NetBT_HPTX1, 08-00-09-DC-0A-A9, VC's: 2, Wan:
Wan
Transport: NetBT_HPTX2, 08-00-09-DB-88-4C, VC's: 0, 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: 17
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: 37,265
SMB's Received: 148
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Bytes Transmitted: 14,970
SMB's Transmitted: 146
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 216,580
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Initally Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 0
Random Read Operations: 0
Read SMB's: 0
Large Read SMB's: 0
Small Read SMB's: 0
Write Operations: 4,165
Random Write Operations: 0
Write SMB's: 0
Large Write SMB's: 0
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 5
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 3
Server Disconnects: 0
Hung Sessions: 0
Use Count: 4
Failed Use Count: 0
Current Commands: 0
Server File Opens: 0
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 0
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: 269
Server Bytes Received: 477
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

NT Registry

Inetinfo:

```
Key Name:          SYSTEM\CurrentControlSet\Services\Inet-
Info
Class Name:        <NO CLASS>
Last Write Time:   1/14/98 - 11:33 AM

Key Name:          SYSTEM\CurrentControlSet\Services\Inet-
Info\Parameters
Class Name:        <NO CLASS>
Last Write Time:   3/9/97 - 2:32 PM
Value 0
  Name:            BandwidthLevel
  Type:            REG_DWORD
  Data:            0xffffffff

Value 1
  Name:            ListenBackLog
  Type:            REG_DWORD
  Data:            0x1770

Value 2
  Name:            MaxPoolThreads
  Type:            REG_DWORD
  Data:            0xa0

Value 3
  Name:            PoolThreadLimit
  Type:            REG_DWORD
  Data:            0xa0

Value 4
  Name:            ThreadTimeout
  Type:            REG_DWORD
  Data:            0x15180
```

```
Key Name:          SYSTEM\CurrentControlSet\Services\Inet-
Info\Parameters\Filter
Class Name:        <NO CLASS>
Last Write Time:   1/14/98 - 11:33 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\Inet-
Info\Parameters\MimeMap
Class Name:        <NO CLASS>
Last Write Time:   1/14/98 - 11:33 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\Inet-
Info\Performance
Class Name:        <NO CLASS>
Last Write Time:   1/14/98 - 11:33 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:            0x738

Value 3
  Name:            First Help
  Type:            REG_DWORD
  Data:            0x739

Value 4
  Name:            Last Counter
  Type:            REG_DWORD
  Data:            0x756

Value 5
  Name:            Last Help
  Type:            REG_DWORD
  Data:            0x757

Value 6
  Name:            Library
  Type:            REG_SZ
  Data:            infoctrs.DLL

Value 7
  Name:            Open
  Type:            REG_SZ
  Data:            OpenINFOPerformanceData
```

tpcc:

```
Key Name:          SOFTWARE\Microsoft\TPCC
Class Name:        <NO CLASS>
Last Write Time:   3/19/97 - 12:54 AM
Value 0
  Name:            BackoffDelay
  Type:            REG_SZ
  Data:            500

Value 1
  Name:            ConnectionPooling
  Type:            REG_SZ
  Data:            OFF

Value 2
  Name:            ConnectionPoolRetryTime
  Type:            REG_SZ
  Data:            0

Value 3
  Name:            DeadlockRetry
  Type:            REG_SZ
  Data:            3

Value 4
  Name:            LastInstalledVersion
  Type:            REG_SZ
  Data:            DBLIB

Value 5
  Name:            LOG
  Type:            REG_SZ
  Data:            OFF

Value 6
  Name:            MaxConnections
  Type:            REG_SZ
  Data:            5000

Value 7
  Name:            MaximumWarehouses
  Type:            REG_SZ
  Data:            1500

Value 8
  Name:            NumberOfDeliveryThreads
  Type:            REG_SZ
  Data:            7

Value 9
  Name:            PATH
  Type:            REG_SZ
  Data:            C:\InetPub\wwwroot\
```

w3svc:

```
Key Name:          SYSTEM\CurrentControlSet\Ser-
vices\W3SVC\Parameters
Class Name:        <NO CLASS>
Last Write Time:   3/7/97 - 2:33 PM
Value 0
  Name:            AcceptExOutstanding
  Type:            REG_DWORD
  Data:            0x1770

Value 1
  Name:            AccessDeniedMessage
  Type:            REG_SZ
  Data:            Error: Access is Denied.

Value 2
  Name:            AdminEmail
  Type:            REG_SZ
  Data:            Admin@corp.com

Value 3
  Name:            AdminName
  Type:            REG_SZ
  Data:            Administrator

Value 4
  Name:            AnonymousUserName
  Type:            REG_SZ
  Data:            IUSR_uwlc1_000000000

Value 5
  Name:            Authorization
  Type:            REG_DWORD
  Data:            0x5

Value 6
  Name:            CacheExtensions
  Type:            REG_DWORD
  Data:            0x1

Value 7
  Name:            CheckForWAISDB
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            ConnectionTimeOut
  Type:            REG_DWORD
  Data:            0x2260

Value 9
  Name:            DebugFlags
  Type:            REG_DWORD
  Data:            0x8

Value 10
  Name:            Default Load File
  Type:            REG_SZ
  Data:            Default.htm

Value 11
  Name:            Dir Browse Control
  Type:            REG_DWORD
  Data:            0x4000001e

Value 12
```

```
  Name:            Filter DLLs
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\sspifilt.dll

Value 13
  Name:            GlobalExpire
  Type:            REG_DWORD
  Data:            0xffffffff

Value 14
  Name:            InstallPath
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv

Value 15
  Name:            LogFileDirectory
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\System32\LogFiles

Value 16
  Name:            LogFileFormat
  Type:            REG_DWORD
  Data:            0

Value 17
  Name:            LogFilePeriod
  Type:            REG_DWORD
  Data:            0x1

Value 18
  Name:            LogFileTruncateSize
  Type:            REG_DWORD
  Data:            0x1388000

Value 19
  Name:            LogSqlDataSource
  Type:            REG_SZ
  Data:            HTTPLOG

Value 20
  Name:            LogSqlPassword
  Type:            REG_SZ
  Data:            sqllog

Value 21
  Name:            LogSqlTableName
  Type:            REG_SZ
  Data:            Internetlog

Value 22
  Name:            LogSqlUserName
  Type:            REG_SZ
  Data:            InternetAdmin

Value 23
  Name:            LogType
  Type:            REG_DWORD
  Data:            0

Value 24
  Name:            MajorVersion
  Type:            REG_DWORD
  Data:            0x2

Value 25
  Name:            MaxConnections
  Type:            REG_DWORD
  Data:            0x2000

Value 26
  Name:            MinorVersion
  Type:            REG_DWORD
  Data:            0

Value 27
  Name:            NTAuthenticationProviders
  Type:            REG_SZ
  Data:            NTLM

Value 28
  Name:            ScriptTimeout
  Type:            REG_DWORD
  Data:            0x384

Value 29
  Name:            SecurePort
  Type:            REG_DWORD
  Data:            0x1bb

Value 30
  Name:            ServerComment
  Type:            REG_SZ
  Data:

Value 31
  Name:            ServerSideIncludesEnabled
  Type:            REG_DWORD
  Data:            0x1

Value 32
  Name:            ServerSideIncludesExtension
  Type:            REG_SZ
  Data:            .stm


Key Name:          SYSTEM\CurrentControlSet\Ser-
vices\W3SVC\Parameters\Script Map
Class Name:        <NO CLASS>
Last Write Time:   1/14/98 - 11:33 AM
Value 0
  Name:            .idc
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\httpodbc.dll


Key Name:          SYSTEM\CurrentControlSet\Ser-
vices\W3SVC\Parameters\Virtual Roots
Class Name:        <NO CLASS>
Last Write Time:   1/14/98 - 11:34 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
```

Tuxedo UBBconfig

configuration file for client1

```
*RESOURCES
IPCKEY123456

DOMAINIDtpcc
MASTERtpcc
MAXACCESSERS512
MAXSERVERS256
MAXSERVICES768
MAXGROUPS256
MAXMACHINES256
MODELSHM
LDBAL  N
SCANUNIT 60
```

```
BLOCKTIME  5
SANITYSCAN180
DBBLWAIT  2
BBLQUERY180


*MACHINES
DEFAULT:
APPDIR="c:\inetpub\wwwroot"
TUXCONFIG="c:\inetpub\wwwroot\tuxconfig"
TUXDIR="c:\tuxedo"

UWLC1LMID=tpcc

*GROUPS
TPCCGROUP
LMID=tpccGRPNO=1OPENINFO=NONE

*SERVERS
DEFAULT:
CLOPT="-A"

tux_serverSRVGRP=TPCCGROUP SRVID=1 RQADDR=tpcc REPLYQ=Y
MIN=85

*SERVICES
NEW_ORDER
STOCK_LEVEL
PAYMENT
ORDER_STATUS
```

RTE Configuration parameters

Driver Startup Parameters and Environment Settings

Run Type: non batch
Run options: -u 13500
Environment:
# configuration file for client1
_=/usr/bin/env
KINDS=HOST SERVER CLIENT CHECKPOINT DELIVERY DRIVER
DATABASE_STATS=0
NR_HOST=1
CHKPNT_INTERVAL2=30
CHECKPOINT_USER=tpcc
MANPATH=/usr/share/man/%L:/usr/share/man:/usr/contrib/man/%L:/usr/contrib/man:/usr/local/man/%L:/usr/local/man:/opt/blinklink/share/man:/opt/ansic/share/man/%L:/opt/ansic/share/man:/opt/langtools/share/man/%L:/opt/langtools/share/man:/opt/dtcmgr/share/man:/usr/omni/man
EMON=\runemon
OS_MENU=0.10
TORNADO_STATS=0
RPT_WINDOW_SIZE=30
SERVER_SQL_OPTIONS=-c -x -t1081 -t3502 -t812
SERVER_MONITOR=\ntreskit\monitor.exe
RESULTS_NAME=run
DRIVER_SPAWN_PERIOD=1
KRATE1=360
STKL_THINK=5.15
SHLIB_PATH=/opt/odbc/drivers:/opt/odbc/lib
VISUAL=/usr/bin/vi
KRATE2=360
TERMKNOWN=yes
PATH=/home/web-tpcc/bin:/home/web-tpcc/scripts:/usr/bin:/opt/ansic/bin:/usr/ccs/bin:/usr/contrib/bin:/opt/nettladm/bin:/usr/bin/X11:/usr/contrib/bin/X11:/opt/langtools/bin:/opt/dtcmgr/sbin:/sbin:/usr/sbin:.:/usr/local/bin:/usr/local/scripts:/usr/bin/X11:/usr/local/bin/X11:/usr/contrib/condor/bin:/usr/lib/SoftWindows/bin
DELIVERYS=uwlc1 uwlc2 uwlc4
NUMBER=1
EMONDUR=420
RESULT_DIR=/home/web-tpcc/results/run.170
DELIVERY_TYPE=iis-web-delivery
CHECKPOINT_TEMP_NAME=/tmp/checkpoint.6202
SERVER_RES_KIT=\ntreskit
NR_DRIVER=6
OUTPUT_DIR=/home/web-tpcc/run_output
CLAST_CONST_C=190
WEB_TPCC=1
COLUMNS=80
CHECKPOINT_TOUCH=\mksnt\touch.exe
SQLSERVR=\mssql\binn\\sqlservr
CLIENT_TUX_DIR=c:\tuxedo
SERVER_USER=tpcc
BATCH_TPCC=false
NR_CHECKPOINT=1
TPCC_SERVICE=80
NR_CLIENT=3
DRIVER_PATH=/home/web-tpcc/bin/hpux-web-driver
CLIENT_CONFINFO=\tools\bin\confinfo.bat
COMM_ADJUST_PMT=0.10
ROUTINE=DRIVER_finish
CLIENT_RTIME_NAME=\temp\rtime.exe
KERNEL_STATS=0
CHECKPOINTS=uwlc2
DVRY_THINK=5.15
CHECKPOINT=uwlc2
SERVER_CP_FILE=schedcp.cmd
SERVER_SC=\ntreskit\sc.exe
SERVER_TYPE=ms-sqlserver
count=1
THIS_TYPE=hpux-web-driver
CHECKPOINT_LOG_PATH=\temp\checkpoint.log
CLIENT_RTIME_FILE=rtime.exe
COMM_ADJUST_NEWO=0.10
DELIVERY_USER=tpcc
EDITOR=/usr/bin/vi
CHECKPOINT_LOG_FILE=checkpoint.log
SERVER_CP_NAME=\temp\schedcp.cmd
NEWO_THINK=12.12
STKL_MENU=0.10
LOGNAME=web-tpcc
SQLISQL=\mssql\binn\isql
DRIVER_AUTO=false
DELIVERY_DELILOG=delilog
DELIVERY_LOGS=logs
CHKPNTDUR=780
TESTENV=/home/web-tpcc/TESTENV
MAIL=/usr/mail/web-tpcc
SECONDS=3
CLIENTS=uwlc1 uwlc2 uwlc4
CLIENT_PATH=/home/web-tpcc/bin/iis-web-tux
POST_STEPS=shutdown stop_performance cleanup audit finish
COPY_ENV=1
HOST_AUTO=false
PMT_MENU=0.10
DRIVER_PROG=driver.web
ERASE=^?
DB_GAMINIT=true
PS1=web-tpcc@uxr3%
INTERRUPTED=false
HOST_PATH=/home/web-tpcc/bin/hpux-driver-host
PEPSI_STATS=0
HOST=uxr3
DATABASE=sqlserver
TRANS_TIME=85
RUN_ID=170
CLIENT_AUTO=false
NR_SERVER=1
NR_LAN=1
NR_DELIVERY=3
RUN_ID_FILE=/home/web-tpcc/run_id
CHECKPOINT_TYPE=ms-sqlserver-checkpoint
SERVER_RTIME_NAME=\temp\rtime.exe
RANDOMIZE_OUTPUT=1
CLIENT=uwlc1 uwlc2 uwlc4
SKIP_IF_INTERRUPTED=true
SERVER_LOG_PATH=\mssql\log\errorlog
CONFIG_FILE=/project/oracle/v7/bench/tpc/tpcc/admin/p_common.ora
RESULTS_ROOT=/home/web-tpcc/results
CONFINFO=\tools\bin\confinfo.bat

SQLSTATS=true
DRIVER=uxr1 uxr2 uxr5 uxr6 uxr7 uxr8
DRIVER_WAIT_FIFO=prospect_wait
COMM_ADJUST_STKL=0.10
SERVER_AUTO=false
THIS_FILE=/home/web-tpcc/bin/hpux-web-driver/routines
CHECKPOINT_SCRIPT=\temp\checkpoint.ksh
SERVER_RTIME_FILE=rtime.exe
SERVER_PATH=/home/web-tpcc/bin/ms-sqlserver
DVRY_MENU=0.10
OUTPUT_PATH=/home/web-tpcc/run_output/output.170
SYSTEM=uxr3
DRIVERS=uxr1 uxr2 uxr5 uxr6 uxr7 uxr8
THIS_PATH=/home/web-tpcc/bin/hpux-web-driver
PERFORMANCE_DETAIL=0
OUTPUT_FILE=output
SERVERS=prf_sut6
GPROFDIR=.
DRIVER_TYPE=hpux-web-driver
BINDIR=/home/web-tpcc/bin
PREV_DIR=/home/web-tpcc
CLIENT_IIS_PORT=80
COUNT_USERS_TIME=1800
DISPLAY=uxr4:0.0
CHECKPOINT_STOP_FILE=/temp/checkpoint.stop
SERVER_PMW_NAME=\temp\tpcc-server.pmw
COMM_ADJUST_DVRY=0.10
DRIVER_PROSPECT=/home/web-tpcc/bin/hpux-web-driver/prospect.B.10.01
NEWO_KEY=18.01
SQL_OPTIONS=-c -x -t1081 -t3502 -t812 -T1140 -Cd1440000 -Cp4500
CHECKPOINT_PATH=/home/web-tpcc/bin/ms-sqlserver-checkpoint
STKL_KEY=2.01
TRANS_TYPE=0
SHELL=/usr/bin/ksh
CLIENT_TUX_APP_DIR=c:\inetpub\wwwroot
CID_CONST_C=498
DB_SIZE=1350
NEWO_MENU=0.10
DELIVERY_DELISRV_PATH=\inetpub\wwwroot\delisrv.exe
SERVER_STARTED_FIFO=server_started
HOST_TYPE=hpux-driver-host
HISTSIZE=50000
SWINHOME=/usr/lib/SoftWindows
PRE_STEPS=initialize startup getconfiguration start_performance
SERVER_PMW_FILE=tpcc-server
ARGS=-u 13500
TRANS_NUM=13000000
KERNRAT2=\runkrate -M -s 180
OUTPUT_LEVEL=3
CHECKPOINT_AUTO=false
HOME=/home/web-tpcc
SQLDIR=\mssql
DELIVERY=uwlc1 uwlc2 uwlc4
CLIENT_TYPE=iis-web-tux
KERNRAT1=\runkrate -K -s 360
CLIENT_USER=tpcc
SERVER=prf_sut6
OS_THINK=10.15
IID_CONST_C=3415
MAILER=/usr/bin/elm
AUDIT=false
HOSTS=uxr3
TERM=vt100
DVRY_KEY=2.01
CPI_STATS=0
HOST_USER=web-tpcc
COLLECT_EMON=false
CLIENT_MONITOR=\ntreskit\monitor.exe
PWD=/home/web-tpcc/results/run.170
BASE_SHMEM_KEY=55555
CLIENT_PERF_LOG_NAME=\temp\tpcc-client.log
OS_KEY=2.01
COMM_ADJUST_ORDS=0.10
CLIENT_COUNT_USER_SLEEP=120
TZ=PST8PDT
PMT_KEY=3.01
CLIENT_PMW_FILE=tpcc-client
CLIENT_RES_KIT=\ntreskit
SAR_STATS=0
SERVER_FSPERF_LOG=\temp\fsperf.txt
DRIVER_USER=web-tpcc
ENV=/home/web-tpcc/.kshrc
SERVER_PERF_LOG_NAME=\temp\tpcc-server.log
PMT_THINK=12.15
DELIVERY_AUTO=false
CHKPNT_INTERVAL=30
DELIVERY_DELILOG_PATH=\inetpub\wwwroot\delilog
DRIVER_SPAWN_RATE=2
DELIVERY_PATH=/home/web-tpcc/bin/iis-web-delivery
CLIENT_PMW_NAME=\temp\tpcc-client.pmw
CLIENT_SC=\ntreskit\sc.exe
ONYXE_STATS=0
LINES=25
START_DIR=/home/web-tpcc
step=initialize
MAILCHECK=300
A__z=-5MAILCHECK

Appendix D – Disk Storage

180-day and 8 hour Space Calculations are provided below:

180 day growth						
No Warehouses	**1350**					
TpmC	**16257.2**					
Table	**Rows**	**Data**	**Index**	**Extra 5%**	**Data+Index+5%**	
Warehouse	1,350	2700	14	135.7	2849.7	
District	13500	27000	112	1355.6	28467.6	
Customer	40500000	27005400	2095990	1455069.5	30556459.5	
History	40500000	2025002			2025002	
NewOrder	12150000	135000	820	6791	142611	
Orders	40500000	1053000	6348		1059348	
OrderLine	405000764	22513178	147152		22660330	
Item	100000	9100	46	457.3	9603.3	
Stock	135000000	45009000	248676	2262883.8	47520559.8	
Total		97779380	2444484.5	3726692.9	103950557.4	
Segment		**Size**				
Master & Msdb & Model & Tempdb		100,000				
Misc_seg		11490000				
OrdIn_seg		37458000				
cs_seg		78798000				
Total		127,846,000				
Dynamic Space		25591180	Sum of Data for order, order_line and history			
Static Space		78359377.4	Sum of all data and index +5 % - dynamic space			
Free Space		23,895,443	Total space allocated to DBMS - dynamic - static			
Daily Growth		4930855.484	Dynamic Space*tpmC/(W*62.5)			
Daily Spread		16499159.37	Free Space - 1.5 * Daily Growth			
			This can be reconfigured to eliminate daily spread			
			(zero assumed)			
180 Day Space		**965913364.6**	Static space + 180*(daily growth + daily spread)			
Disk Allocation		**Space Needed**	**Disk Size**			**Disks Needed**
180 day space		965,913,365	4,160,512	RAID 0		48
			8,885,248	RAID 0		96
8 hr log space		42,878,460	8,885,248	RAID 1		10
OS and swap		5,000,000	8,885,248	RAID 0		1
Note: Numbers are in Kbytes unless otherwise specified						

8 hour space		
Warehouses	**1,350**	
TpmC	**16,257.20**	

BEFORE

Data pages in Syslogs	Before	15,195,284
Table name	Data Space	Index Space
History	2,311,688	0
Orders	1,218,788	12,728
Order_line	25,792,316	202,204
New Order	46,416,488	

AFTER

Data pages in Syslogs	After	17,110,366
Table name	Data Space	Index Space
History	2,345,478	0
Orders	1,236,678	12,728
Order_line	26,179,324	206,464
New Order	47,113,540	

Table	Space Before	Space After	Space Used	# of New Order Trns	Pages per New Order Trn	8 Hour Space
Syslogs	30,390,568	34,220,732	3,830,164	697,052	2.75	**42,878,460**
History	2,311,688	2,345,478	33,790	697,052	0.02	378,277
Orders	1,231,516	1,249,406	17,890	697,052	0.01	200,277
Order_line	25,994,520	26,385,788	391,268	697,052	0.28	4,380,222

Note: All numbers are in kbytes unless otherwise specified

Appendix E – Quotations

All quotes can be found on the following pages.

Software House International **Pricing Proposal**	**Quotation #MO-980327-31840** **03/27/98**

Hewlett Packard
Larry Gray
Netserver Division

Phone: 408-343-6288 **Fax: 408-343-6755**

SHI Account Exec: Matthew O. Martin
Telephone : (800) 766-6357
Fax : (408) 526-1222

Reference: This quote good for 60 days.

Product	Part #	Qty	List	Your Price	Total
SCSI Cable .09m	C2911A	4		$124.00	$496.00
Mylex EDAC 32MB	Dacsxi5w32-	5		$3295.00	$16,475.00
HP 9.1GB 10000 RPM Drive	D6019A	12		$1675.00	$20,100.00
HP Rack Storage/8	D4902A	22		$2100.00	$46,200.00
HP Storage System/8	D3604B	5		$1015.00	$5,075.00
HP 9gb SCSI-2	D4289A	106		$1229.00	$130,274.00
HP 4.2gb SCSI-2	D3583C	53		$721.00	$38,213.00
HP 9gb SCSI-2 Tray	D4911A	1		$1260.00	$1,260.00
HP Surestore 6000I	C1528F	1		$905.00	$905.00
Microsoft NTSever 4. Ent	779-00001	1		$3345.00	$3,345.00
Microsoft SQL 6.5 Ent.	810-00007	1		$24410.00	$24,410.00
Compex Microhub	MX1205	3		$189.00	$567.00
Netlux 16 Port	NX-H16EZ	990		$75.00	$74,250.00
Netserver LC II	D5014A	3		$2375.00	$7,125.00
P266 CPU Upgrade	D4995A	3		$924.00	$2,772.00
Total					$371,467.00

Additional Comments:

Software House International
Pricing Proposal

Quotation #MO-980327-31840
03/27/98

Hewlett Packard
Larry Gray
Netserver Division

Phone: 408-343-6288 Fax: 408-343-6755

SHI Account Exec: Matthew O. Martin
Telephone : (800) 766-6357
Fax : (408) 526-1222

Reference: This quote good for 60 days.

Product	Part #	Qty	List	Your Price	Total
Netserver LXR Pro8	D5028A	1		$28260.00	$28,260.00
NetSever LX Dual Processor Upgrade	D5030A	3		$11558.00	$34,674.00
HP 10/100 PCI Card	J3171A	1		$72.00	$72.00
14 in HP Monitor	D2821S	4		$205.00	$820.00
HP Netserver Key/mouse	C3751/47/b	1		$126.00	$126.00
APC Smart-UPS 3000NS	588293	1		$1740.00	$1,740.00
2.0 Meter Rack	J1487B	2		$1470.00	$2,940.00
Netserver Rack Instal Kit	D4984B	1		$54.00	$54.00
5Yr Support Netserver	H5526A	1		$2050.00	$2,050.00
256MB HP ECC DIMM	D5026A	15		$2310.00	$34,650.00
DACPJ-3-8E SCSI Ctlr 3ch	Z75090	8		$1555.00	$12,440.00
64MB Admor Upgrd for above	ADC64-DAC	6		$159.00	$954.00
NetRaid RS/8	D4983A	9		$86.00	$774.00
Ext. Scsi Cable	D3637C	14		$86.00	$1,204.00
SCSI Cable 2.5m	D3638C	9		$86.00	$774.00
Total					**$121,532.00**

Additional Comments:

Software House International
Pricing Proposal

Quotation #MO-980327-31840
03/27/98

Hewlett Packard
Larry Gray
Netserver Division

Phone: 408-343-6288 Fax: 408-343-6755

SHI Account Exec: Matthew O. Martin
Telephone : (800) 766-6357
Fax : (408) 526-1222

Reference: This quote good for 60 days.

Product	Part #	Qty	List	Your Price	Total
128MB Dimms	D4297A	12		$688.00	$8,256.00
Netserver 10/100 NIC	D5013A	15		$83.00	$1,245.00
Warranty upgrd for D5014A	H5519A	3		$608.00	$1,824.00
Windows NT Server	ZS750001	3		$475.00	$1,425.00
SQL SVR Prog. Toolkit	ZS750002	1		$129.00	$129.00
Microsoft C++ V4.0	ZS750003	1		$425.00	$425.00
Power Distribution Unit	E7875A	4		$169.00	$676.00
Power Cords	E7802A	4		$46.00	$184.00
10/100 NIC	J3171A	1		$72.00	$72.00
Total					**$14,236.00**

Microsoft

March 19, 1998

Mr. Larry Gray
Product Manager
Hewlett-Packard Company
Network Server Division
5301 Stevens Creek Blvd.
Santa Clara, CA 95052

via FAX # 408-343-6755

Dear Larry,

Microsoft has received your request for permission to disclose the results of TPC-C benchmark tests conducted by HP with Microsoft SQL Server, Enterprise Edition 6.5 on the following system:

HP NetServer LX Pro, 8 processors, Pentium Pro, 200 MHz, 1MB cache
Test Results: 16000 tpmC @ $35/tpmC approx.

Microsoft hereby grants HP permission to disclose these results to third parties and acknowledges that HP has formally requested permission to do so in accordance with the license agreement for Microsoft SQL Server6.5, Enterprise Edition software.

Best regards,

Sid Arora
Product Manager, Microsoft SQL Server
Applications and Tools Group

BEA ENTERPRISE MIDDLEWARE SOLUTIONS

March 16, 1998

Hewlett Packard
Cupertino, CA

Dear Mr. Larry Gray:

Per your request I am enclosing the pricing information regarding TUXEDO 6.x that you requested. This pricing applies to Tuxedo 6.1, 6.2, 6.3 and 6.4. Please note that Tuxedo 6.4 is our most recent version of Tuxedo but that all 6.x releases are generally available. Core functionality services pricing is appropriate for your activities. As per the table below, HP server systems are classified in one of 5 tiers based on CPU type and capacity.

Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.x. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

BEA Tux/CFS Unlimited User 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, entry level RISC Uni-processor workstations and servers (Class 1and Class 2)	Unlimited	$3,000.00	$450.00	$660.00
Tier 2 -- PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations (class 3)	Unlimited	$12,000.00	$1,800.00	$2,640.00
Tier 3 -- Midrange Multiprocessors, up to 8 CPUs per system capacity (Class 4 and 5)	Unlimited	$30,000.00	$4,500.00	$6,600.00
Tier 4 -- Large (more than 8, less than 32 CPUs) and Mainframe Systems (Class 6)	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

HP 9000 server tier classifications

Platform	Tier 1	Tier 1	Tier 2	Tier 3	Tier 3	Tier 4	Tier 5
Hewlett-Packard	Uni-processor Workstations	9000/E25 9000/E35 9000/E45 9000/E55 9000/G30 9000/G40	9000/G50 9000/G60 Multi-Processor Workstations 9000/D200, 210 220,230	9000/H20 9000/H30 9000/H40 9000/H50 9000/I30 9000/I40 9000/K1XX 9000/D310, 320,330 9000/D250/60	9000/I50,60 9000/H60 9000/G70 9000/H70 9000/I70 9000/K2XX 9000/K3XX 9000/K4XX 9000/K5XX 9000/D350/60/70	9000/T500, T600 1-16 CPUs	9000/V series all models

Intel based server tier classifications:

Platform	Operating System	Tier 1	Tier 1	Tier 2	Tier 3	Tier 3
Intel Pentium/ Pentium Pro PCs	Interactive R3.2 ESIX SVR 4.0 SCO UNIX 3.2.2 and 3.2.4 SCO ODT 2.x,3.x Solaris x86 2.X UnixWare, Windows NT 3.5/4.0	All 386/486 PCs are Class 1	ALL Pentium and Pentium Pro PCs with 1 or 2 CPUs capacity are Tier 1	ALL Pentium and Pentium Pro PCs with 3 or 4 CPUs capacity are Tier 2		ALL Pentium and Pentium Pro PCs with 5,6,7, or 8 CPUs are Tier 3

Very Truly Yours,

Lewis D. Brentano,
Director, Market Planning
BEA Systems, Inc.