Hewlett-Packard Company
Network Server Division

HP NetServer LX Pro
Using Microsoft SQL Server 6.5 on Microsoft NT 4.0

TPC Benchmark® C
Full Disclosure Report

Second Edition
September 9, 1997

First Edition - September 9, 1997 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., September 9, 1997

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the NetServer LX Pro 6/200 SMP 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. 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-vi*** contain the executive summary of the benchmark results for the HP NetServer LX Pro 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 6/200 LX Pro SMP (4-way)	Microsoft SQL Server v6.5 Enterprise Edition	Microsoft NT Server 4.0 Enterprise Edition
Availability Date — HW: current SW: 11/30/97			

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® ($463,178/9198.37)
$463,178	**9198.37 tpmC®**	**$50.35 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

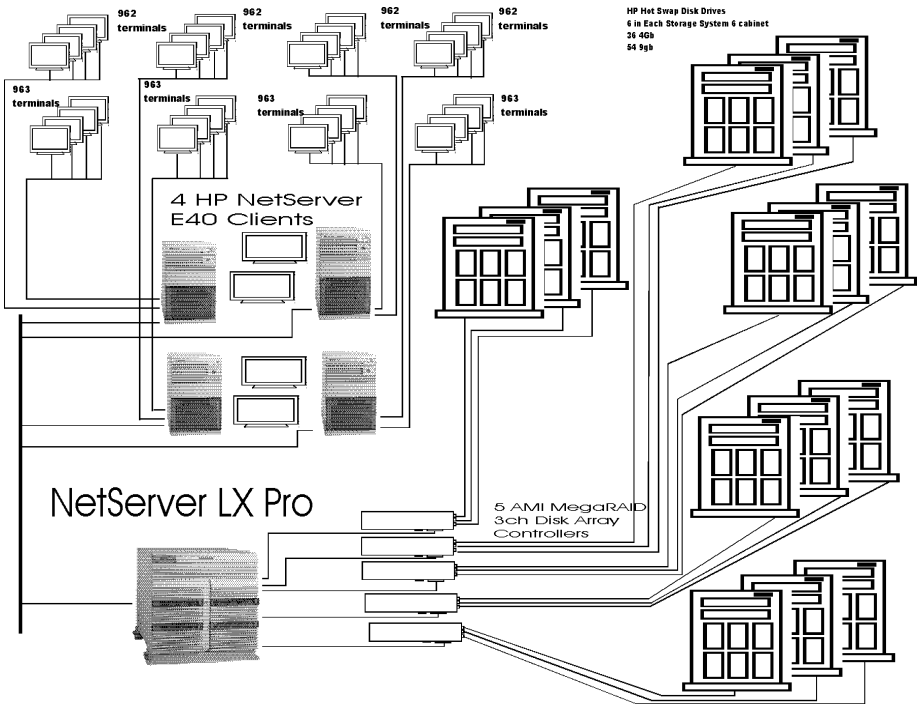
or

Hewlett-Packard Company/NetWork Server Division
10450 Ridgeview Court
PO Box 4050, MS 49EL-FS
Cupertino, CA 95015-4050 USA

attention: Jim Nagler

phone: 408 343-6332

Hewlett-Packard Co.	NetServer LX Pro 6/200 SMP Client/Server with 4 NetServer E-40 front-ends			TPC-C® Rev 3.3 Report Date: September 9, 1997
Total System Cost	**TPC-C® Throughput**		**Price/Performance**	**Availability Date**
$463,178	**9198.37 tpmC®**		**$50.35 / tpmC®**	**November 30, 1997**
Processors	**Database Manager**	**Operating System**	**Other Software**	**Number of Users**
4 Intel Pentium Pro 200MHz	**Microsoft SQL Server v. 6.5 Enterprise Edition**	**Microsoft NT Server 4.0 Enterprise Edition**	**Microsoft C++ Compiler TUXEDO Transaction Monitor**	**7,850**



System Components	*Server (LX Pro)*		*Each Client (E40)*	
	Qty	*Type*	*Qty*	*Type*
Processors	**4**	**200 MHz Intel Pentium Pro**	**1**	**200 MHz Intel Pentium Pro**
Cache memory	***each***	**512KB cache**		**256 KB cache**
Memory	**4**	**GB**	**256**	**MB**
Disk Array Controllers	**5**	**HP NetRaid Controllers**	**1**	**HP SCSI-2 Controller**
Disk Drives	**60** **36**	**HP Hot-Swap 9G SCSI** **HP Hot-Swap 4G SCSI**	**1**	**2 GB Disk**
Total Storage (GB)	**684**	**GB**		
Tape Drives	**1**	**DAT Storage System**		
Terminals	**1**	**Console terminal**	**1**	**Console terminal**

Hewlett-Packard Co.	NetServer LX Pro 6/200 SMP Client/Server with 4 HP NetServer E40 front ends	TPC-C® Rev 3.3 Report Date: September 9, 1997

Hardware and Software Pricing

Description	Part Number	3d Party Brand	3d Party $	Unit Price	Qty.	Extended Price	5 Year Maint. $
Server Hardware							
HP NetServer LX Pro 6/200 SMP M1 Array Dual Pentium Pro	D4311B	HP	1	11,305	1	11,305	
3 year, on-site, next day warranty	included						
2 200MHz Pentium Pro processors, 64MB memory	included						
12 Hot swap disk bays	included						
Integrated dualPCI F/W SCSI-2cntrlrs&IDEcontrller	included						
Integrtd1024x768 16color 512KB video memory	included						
CD-ROM drive, 3.5-inch, 1.44MB floppy drive	included						
1 parallel, 2 serial ports, keyboard & mouse	included						
3 410W hot-swap power modules & redundant fans	included						
HP NetServer Navigator, HP NetServer Assistant	included						
HP Open View for Windows.	included						
NetServer LX 6/200 Dual Processor card	D4866A	HP	1	825	1	825	
NetServer LX 6/200 Pentium Pro chip	D4867A	HP	1	2,145	3	6,435	
4GB memory board with 512MB	D4967A	HP	1	8,025	1	8,025	
512 MByte ECC DIMM Memory Module	D4968A	HP	1	8,350	7	58,450	
SupportPack-NetServer(upgd wrnty to same day,4hr)	H5520A	HP	2				5,396
14" VGA Monitor	D2813A	HP	1	245	1	245	168
HP 10/100 PCI Network Adaptor	J3171A	HP	1	72	1	72	
Server Storage	**Server Hardware Subtotal:**					**85,357**	**5,564**
AMI MegaRAID Ultra GT F/W SCSI Ctlr 3chnl inc. spare	432110000	AMI	1	1,795	6	10,770	
InternalSCSIcable(int68-pin-ext68pin)+10% spare	5182-6740	HP	1	30	4	120	
ExternalSCSIcable(ext68pin-to SS/6) + 10% spare	5182-6737	HP	1	55	17	935	
HP Storage System/6 + 10% spare	D3604A	HP	1	995	17	16,915	
HP 9GB SCSI-2 hot swap drives + 10% spares	D4289A	HP	1	2,045	66	134,970	
HP 4.2 GB SCSI-2hot swap drive+10% spares	D3583B	HP	1	1,140	40	45,600	
HP 9GB SCSI-2 disk drive (common tray)	D4911A	HP	2	1,450	1	1,450	
HP SureStore Tape 6000i 8Gb Internal DAT	C1528F	HP	1	905	1	905	
Server Software	**Server Storage Subtotal:**					**211,665**	
Microsoft Windows NT Server 4. Enterprise Edition0		MS	3	3,999	1	3,999	
Microsoft SQL Server 6.5 Enterprise Edition		MS	3	24,300	1	24,300	10,475
Client Hardware	**Server Software Subtotal:**					**28,229**	**10,475**
HP Netserver E40 incl 2.1GB disk & 10/100 NIC	D4933A	HP	1	1,885	4	7,540	
64MB SIMMs	D4290A	HP	1	865	16	13,840	
HP 10/100 PCI Network Adaptor	J3171A	HP	1	72	8	576	
14" VGA Monitor	D2813S	HP	1	245	4	980	
5 Yr Support - 14" VGA Monitor		HP	2				672
5 Yr Support - E40 SPU w/internal components	H5517A	HP	2				5,144
Client Software	**Client Hardware Subtotals:**					**22,936**	**5,816**
Windows NT 4.0 Server	Z50001	MS	1	587	4	2.348	
SQL Svr Programmer's toolkit	Z5002	MS	1	129	1	129	
Tuxedo Run TIme		BEA	4	3,000	4	12,000	9,000
Tuxedo Developers Kit		BEA	4	2,495	1	2,495	1,871
Microsoft C++ v. 4.0	Z5003	MS	1	425	1	425	
Communications	**Client Software Subtotals:**					**17,397**	**10,871**
Allied 24-Port 10Base-T EtherNet Hubs + spares	347465		1	179	362	64,798	
	Communications Subtotals:					**64,798**	
				Total		**430,452**	**32,726**

Notes:	**Five Year Cost:**	**463,178**
1.Support pack H5520A upgrades NetServer 3yr warranty to same day, 4hr response, includes SPU internal components (CPUs, mem, NIC, DAT, etc). 2. NetServer support for yrs 4-5 via HP contract support 02A includes all HP components in SPU, $7 per month for monitors	**tpmC Rating:**	**9198.37**
3. Pricing Key: 1=Software House, 2=HP Corporate Price List, 3=Microsoft, 4=BEA Systems	**$/tpmC:**	**50.35**

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.

Note: audited by Richard Gimarc of Performance Metrics, Inc.,

Numerical Quantities Summary for NetServer LX Pro 6/200 SMP

MQTH, Computed Maximum Qualified Throughput **9198.37 tpmC®**

Response Times (in seconds)	90th %-ile	Maximum	Average
New-Order	1.69	4.77	1.18
Payment	1.55	4.58	1.04
Order-Status	2.20	5.22	1.51
Delivery (interactive portion)	1.22	3.98	0.72
Delivery (deferred portion)	0.85	6.00	0.65
Stock-Level	3.85	7.44	2.56
Menu	1.3	5.42	0.69

Transaction Mix, in percent of total transactions

New-Order	44.88%
Payment	43.00%
Order-Status	4.04%
Delivery	4.05%
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	18.10	0.01	12.10	169.09
Payment	3.01	3.02	3.10	0.01	12.15	140.37
Order-Status	2.01	2.02	2.09	0.01	10.16	134.27
Delivery (interactive)	2.01	2.02	2.09	0.01	5.18	49.60
Stock-Level	2.01	2.02	2.09	0.01	5.16	47.53

Numerical Quantities Summary for NetServer LX Pro 6/200 SMP, continued

Test Duration

Ramp-up time		67.6 minutes
Measurement interval		30 minutes
Transactions during measurement interval		614,877
Ramp down time		22.4 minutes

Checkpointing

Number of checkpoints in measurement interval		1
Checkpoint interval		30 minutes

Reproducibility Run

Reproducibility Run	9152.07 tpmC	-0.5%

Preface

This is the full disclosure report for a benchmark test of the NetServer LX Pro 6/200 SMP using Microsoft SQLServer 6.5. 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 LX Pro 6/200 Model 1 server. 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 and the database was Microsoft's SQL 6.5 (253).

The architecture of the NetServer LX Pro was designed by Hewlett-Packard and based on the Intel Pentium Pro chip and associated chipset. The LX used in this test was powered by four 200 MHz Intel Pentium Pro(R) processor chips, each with 512K bytes of SRAM 2nd level cache. In the LX, two 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 7 512 Mbyte DIMMs to the 512 Mbyte DIMM that came with the memory board. This RAM was attached to the system bus via a controller.

This configuration also used two SCSI-2 Fast/Wide PCI Disk controllers that were embedded onto the motherboard and 5 HP AMI 3-channel PCI Disk Array Controllers (DACs). These cards plugged into PCI slots on the motherboard, which are connected to two separate 33MHz PCI I/O buses. Both PCI busses attached directly to the P6 bus through separate PCI bridges so that PCI bus masters can have direct access to memory.

One HP 9Gbyte SCSI-2 (common tray) Fast hard disk and one CDROM drive were attached to one of the embedded PCI SCSI controllers. This disk drive was used exclusively for the Operating System (NT v4.0) and swap space.

In the measured configuration, three pairs of the 9 Gbyte HP SCSI-2 Hot Swap hard disks attached each of the embedded PCI SCSI controllers were used exclusively for the database log. 36 HP 4Gbyte Hot Swap drives and 54 9G Hot Swap disk drives were equally distributed across the 5 3-Channel AMI PCI Disk Array Controllers (DACs). Six Hot Swap disks were assigned per DAC SCSI channel. Each channel was striped and the channels spanned using the AMI Utility. Controller write-back caching and read ahead were specifically disabled.

At the operating system, NT's disk administrator shows 17 disk drives - the 9Gbyte SCSI-2 boot drive, the mirrored sets of 9Gbyte Hot Swap drives used for the log, three156,179 megabyte disk drives externalized by the AMI MegaRAID controller containing the 9.1 Gbyte disks, and two 73,156 Gbyte disks that are externalized by the MegaRAID controllers with the 4Gbyte disk drives. Each of these 2 73Gbyte logical drives represent a hardware stripe set of 18 4Gbyte Hot Swap drives, created at the DAC level with channel spanning. The 156G disks are hardware striped in the same manner using the 9G disks.

The five logical disk drives were used to hold the TPC database. This was done to maximum 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 PCI network adaptor card, attached to the LX motherboard via the PCI bus. This network adaptor supplied a 10BaseT network interface to the four NetServer clients. Each of the clients had 256Mbytes of RAM, one 2Gbyte SCSI hard disk, one HP J3171A PCI network adaptor card, and was running NT 4.0. HP VGA displays were used on the NetServer LX Pro and on each of the four 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 definition 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 Appendix 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 identical to the priced one, except for the addition of 2 4GB and 2 9GB Hot Swap disks for the log growth space and the exchange of 18 9GB disks for 18 4GB disks for the 180 day growth space.

FIGURE 1-1: NetServer LX Pro - Measured Configuration

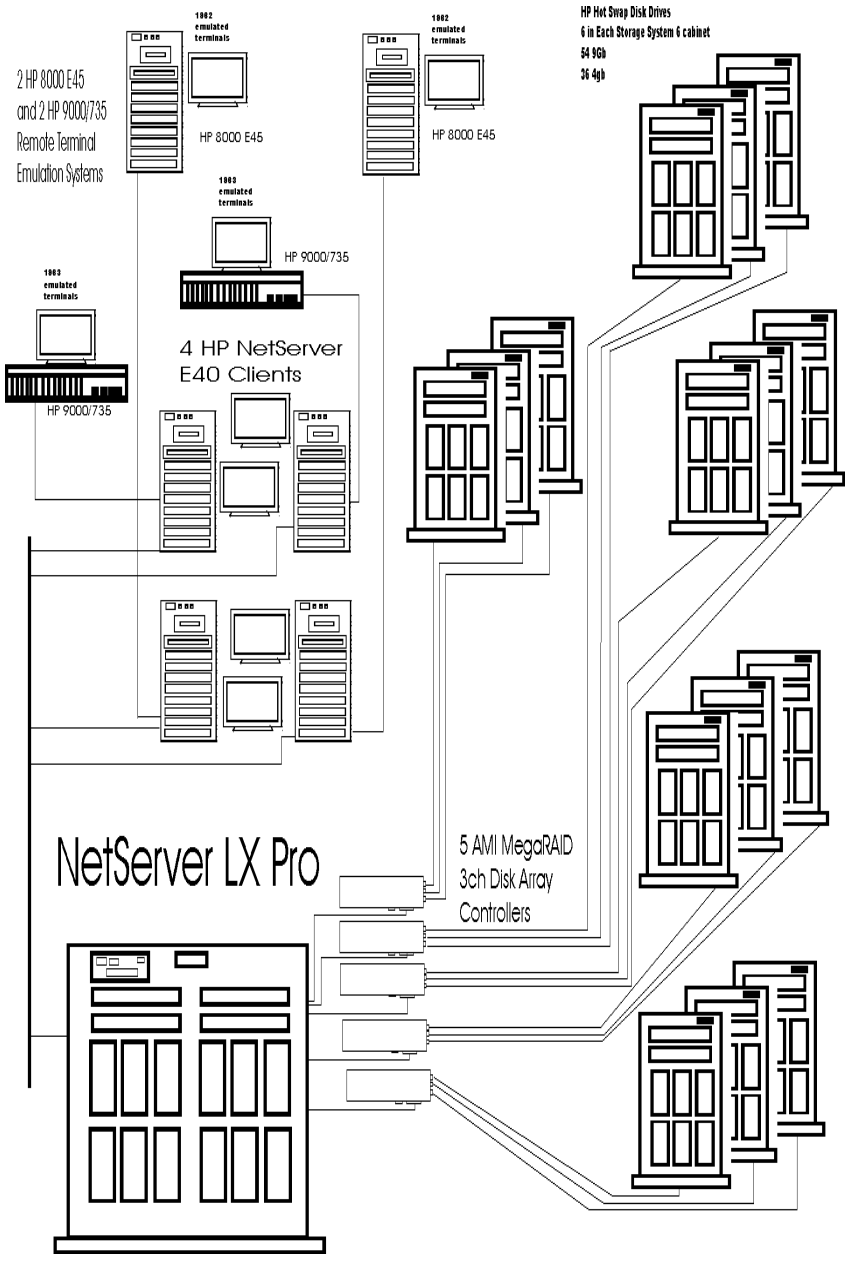
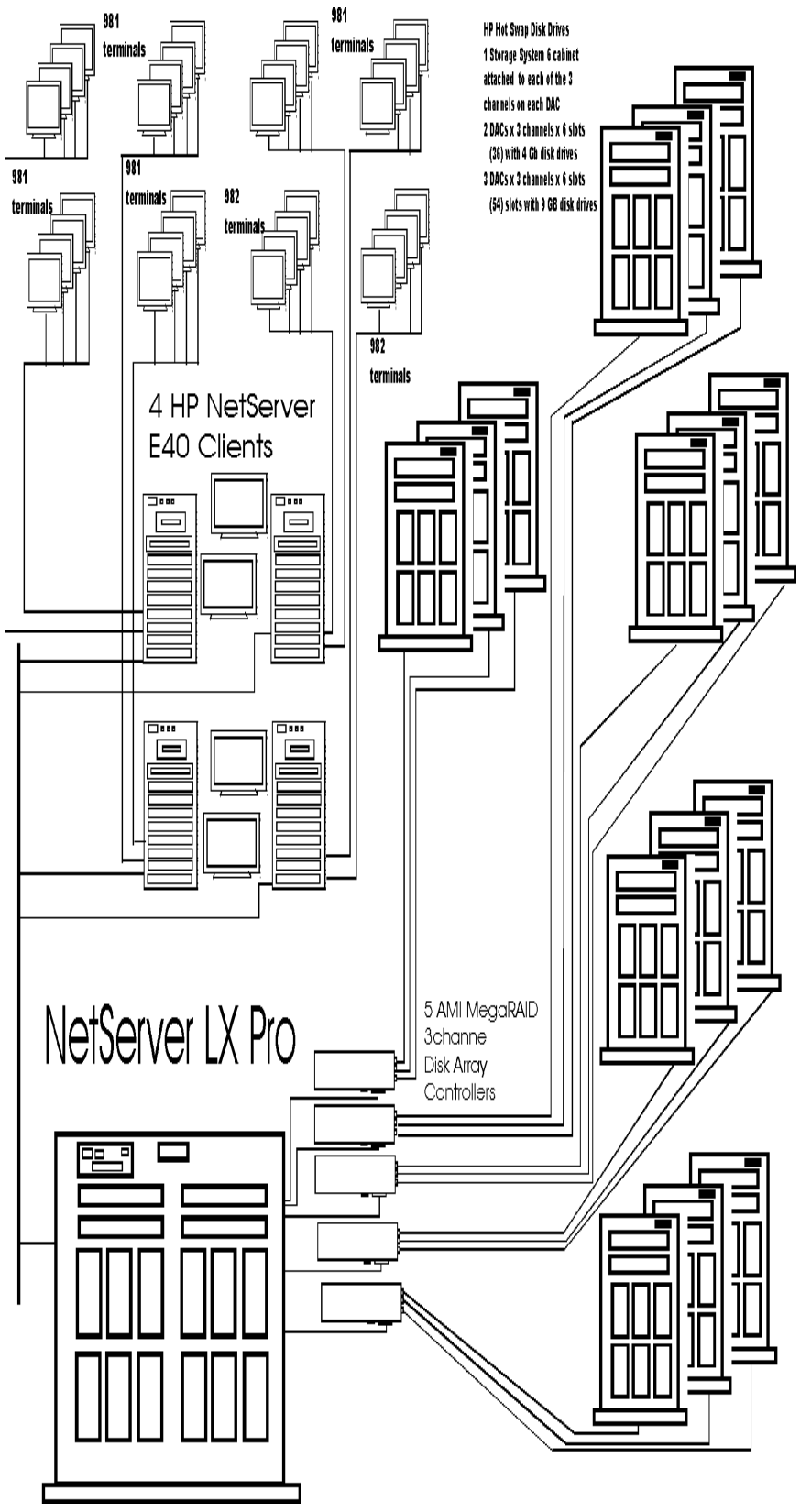


FIGURE 1-2: NetServer LX Pro - Priced Configuration



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.

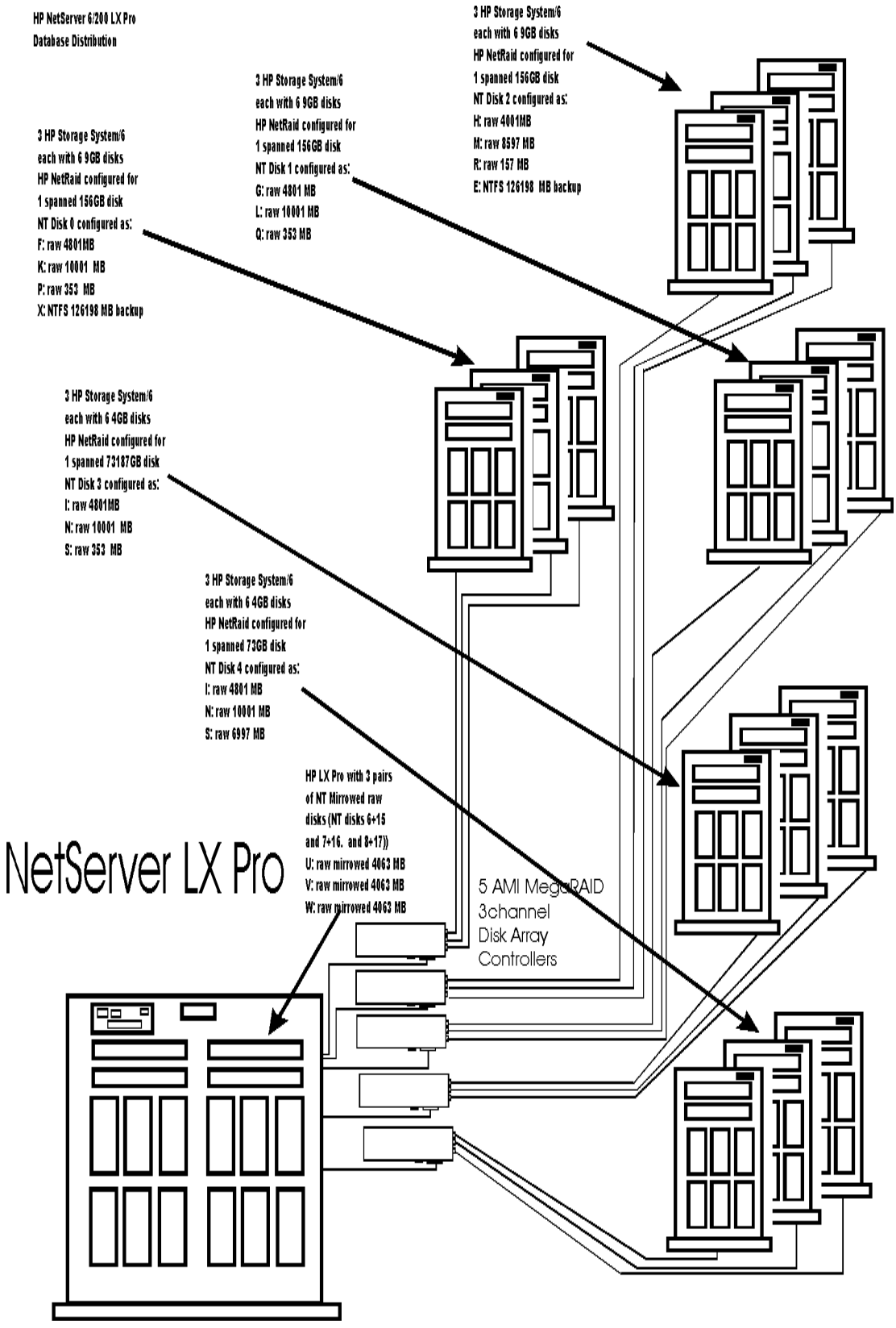
Appendix B contains the code used to define and load the database tables.

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), 60 9Gb, and 36 4Gb Hot Swap disk drives. Figure 3-1 below depicts the data distribution of the files across the hard drives of the HP NetServer LX Pro.

FIGURE 3.1: HP NetServer 6/200 LX Pro Database Distribution



2.3 Insert and Delete Operations

It 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 disclosed. 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.

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

3.2 Input/Output Screen Layout

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%
	Remote Warehouse	14.97
	Non-primary key access	59.88
Order Status	Non primary key access	59.97
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.88%
	Payment	43.00%
	Order Status	4.04%
	Delivery	4.05%
	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 same TUXEDO mechanism that other transactions used. The only 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 LX Pro. 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 LX 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, NT 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. Two 4GB and 2 9GB additional Hot Swap disks were attached to the measured configuration but were not used during the benchmark.

Both configurations used two SCSI-2 Fast/Wide PCI Disk controllers that were embedded onto the motherboard and 5 AMI MegaRAID 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 6 9GB HP SCSI-2 Hot Swap hard disks are used to supply growth space for the log. These are distributed with the first 3 attached to the first internal PCI SCSI controller (A) and the remaining 3 (mirrored pairs of the first 3) attached to the second embedded SCSI controller. 54 HP 9Gbyte Hot Swap drives are attached to 3 of the PCI Disk Array controllers and 36 HP 4Gbyte Hot Swap drives are attached to the other two controllers. Six Hot Swap disks were placed in each HP Storage System 6. Each channel was striped using the MegaRAID 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 17 logical disks - the 9Gbyte SCSI-2 boot drive (common tray), the six mirrored drives used for the log, two 73GB logical drive disks and three 156 GB logical drive disks. Each of these 73GB logical drives represent a hardware stripe set of eighteen 4Gbyte Hot Swap drives, created at the DAC level spanning the three channels. The 156Gbyte 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. The preceding Figure 3-1 depicts the data distribution of the files across the hard drives of the HP NetServer LX Pro.

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	820
District	8,200
Customer	24,600,000
History	24,600,000
Orders	24,600,000
New Orders	7,380,000
Order Line	24,600,000
Stock	82,000,000
Item	100,000

35 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 dynamic 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 24.1GB (including mirror) to sustain the log for 8 hours. Space in the measured and priced configurations available on the transaction log was 26.6GB (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®	9198.37

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	1.18	4.77	1.69
Payment	1.04	4.58	1.55
Order-Status	1.51	5.22	2.20
Interactive Delivery	0.72	3.98	1.22
Deferred Delivery	0.65	6.00	0.85
Stock-Level	2.56	7.44	3.85
Menu	0.69	5.42	1.3

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	18.10
Payment	3.01	3.02	3.10
Order-Status	2.01	2.02	2.09
Interactive Delivery	2.01	2.02	2.09
Stock Level	2.01	2.02	2.09

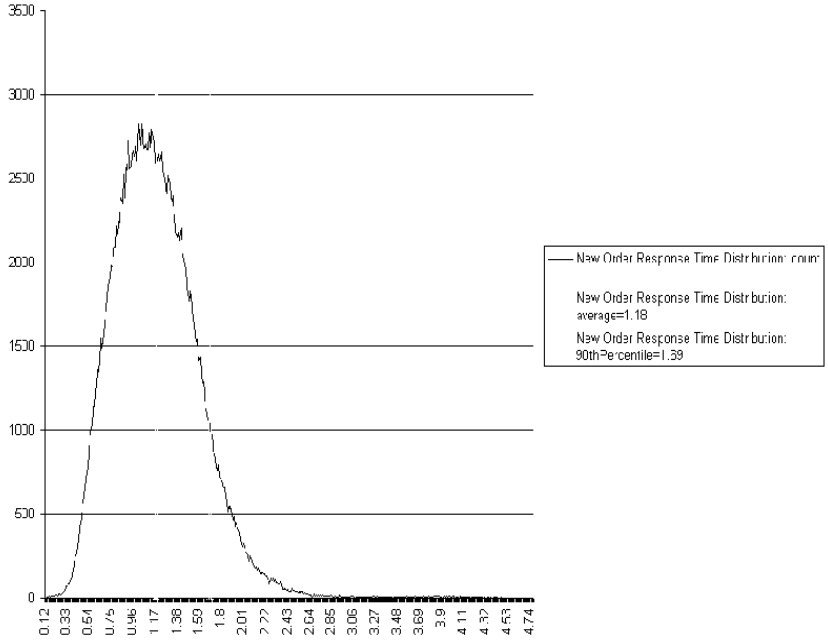
Table 6.4: Think Times

Type	Minimum	Average	Maximum
New-Order	0.01	12.10	169.09
Payment	0.01	12.15	140.37
Order-Status	0.01	10.16	134.27
Interactive Delivery	0.01	5.18	49.60
Stock-Level	0.01	5.16	47.53

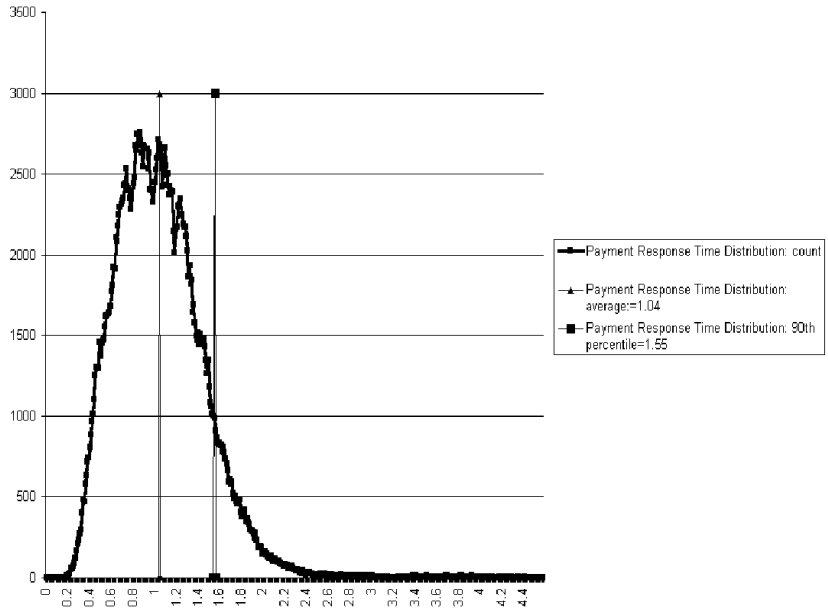
6.4 Response Time Frequency and

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.

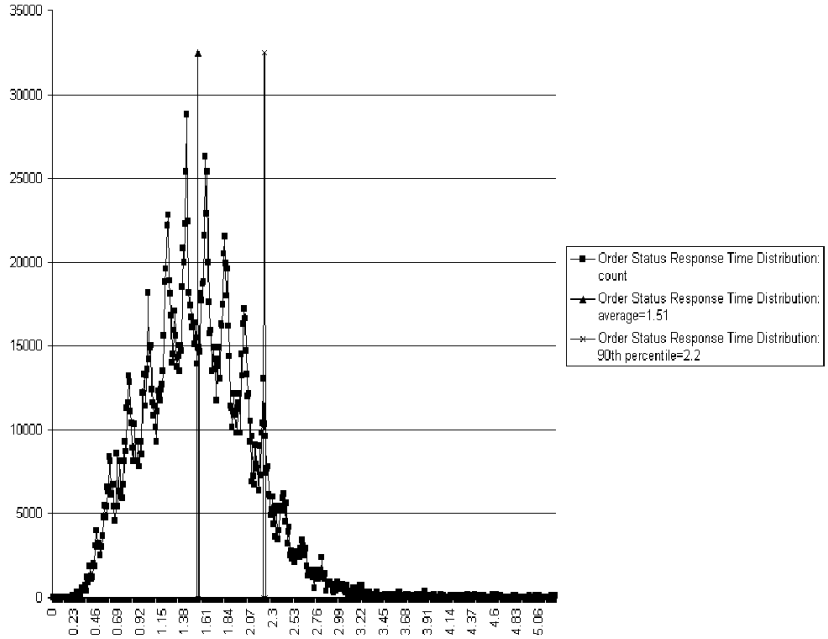
6.4.1 New Order Response Time



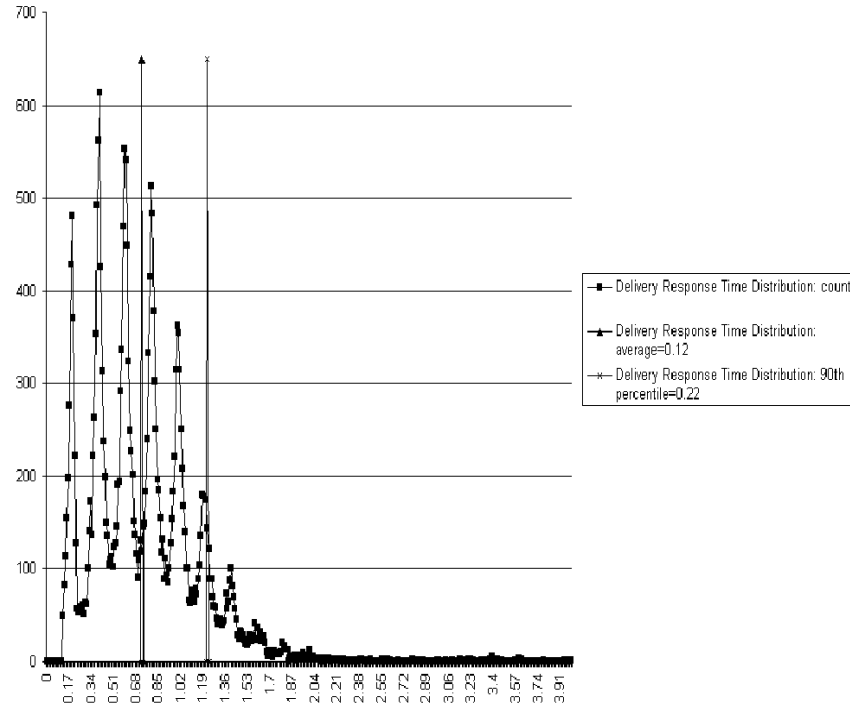
6.4.2 Payment Response Time Distribution



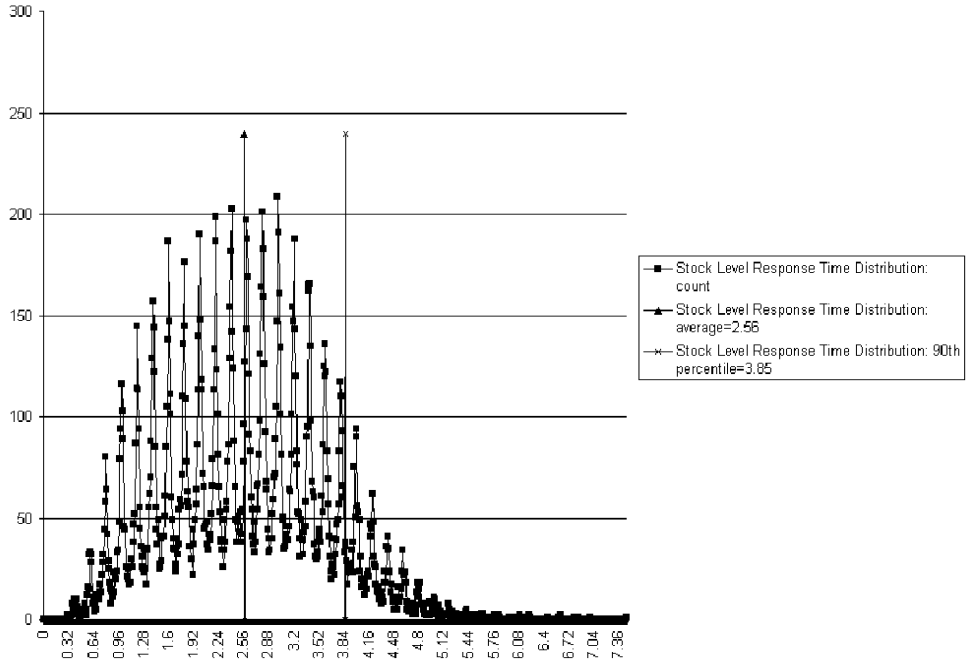
6.4.3 Order Status Response Time



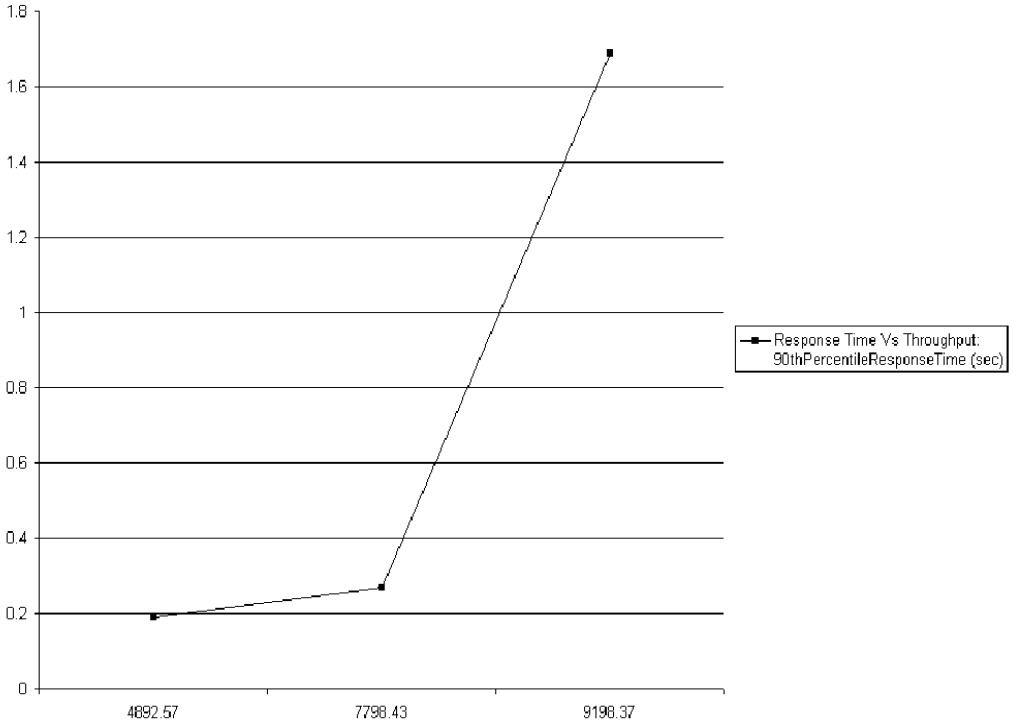
6.4.4 Delivery Response Time Distribution



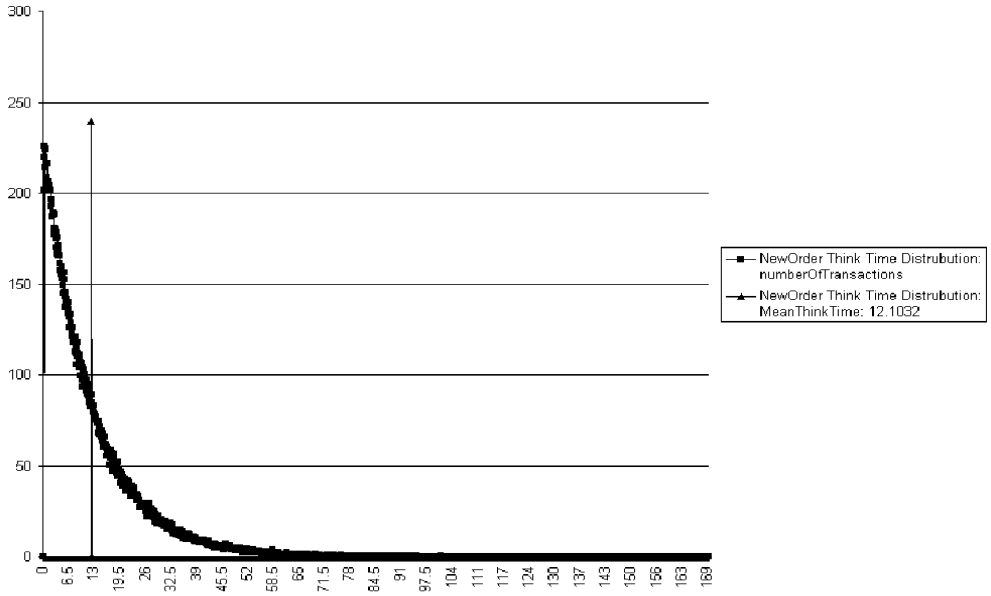
6.4.5 Stock Level Response Time



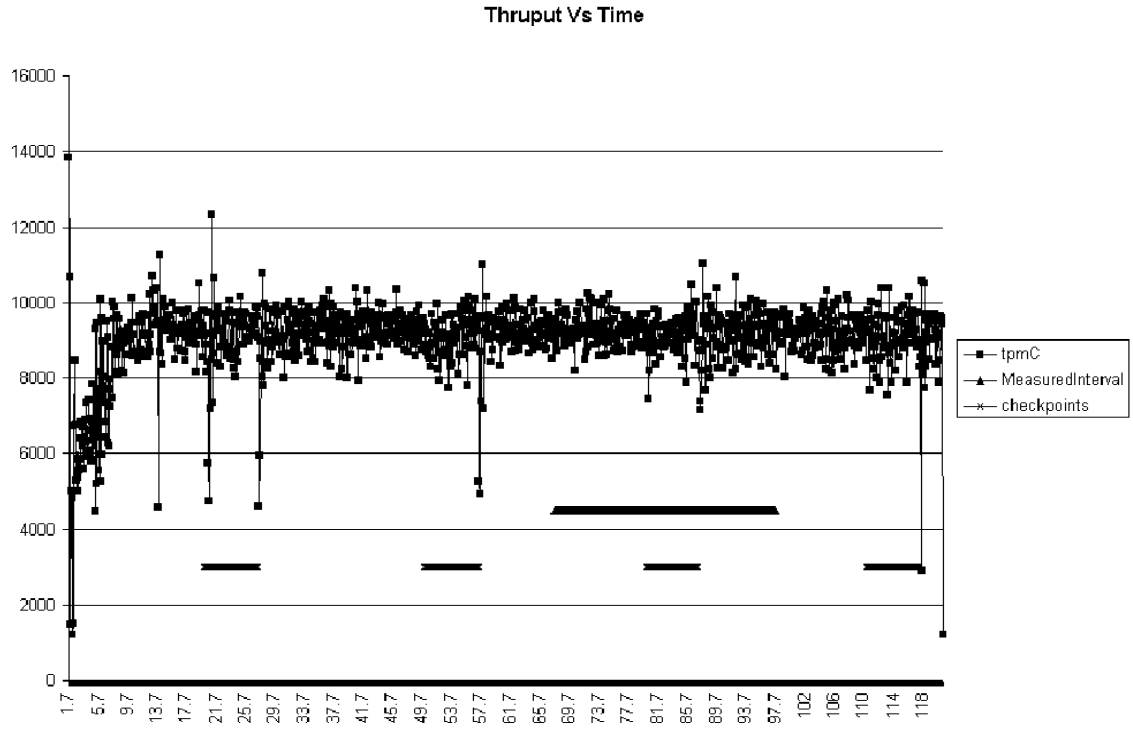
6.4.6 Response Time Versus Throughput



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 9152.97 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	44.88%
Payment	43.00%
Order-Status	4.04%
Delivery	4.05%
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 907 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	1200	1635	435
checkpoint	3000	3429	429
measured interval	4056	5856	1800
checkpoint	4800	5227	427
checkpoint	6600	7032	432

Section 7.0 – Clause 6 Related Items

7.1 RTE description

If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of that input (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 commercially available.

For this instance of the TPC-C benchmark, four driver and four client systems were used. The drivers emulated 7850 users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in figure 7.1

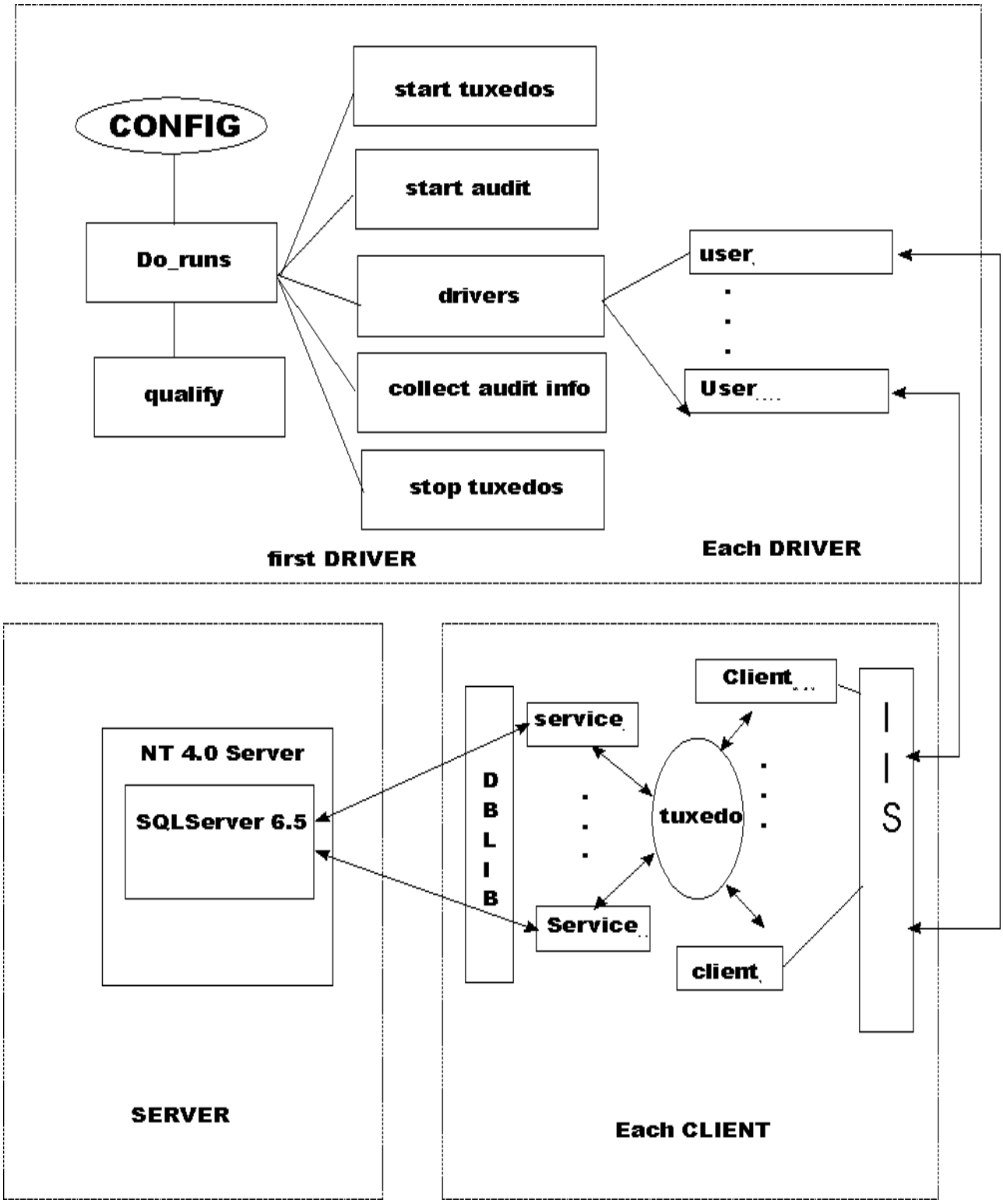
The benchmark is started with the do_runs command on the driver system. do_runs controls the overall execution of the benchmark. After reading a configuration file, do_runs starts the TUXEDO servers on the clients, collects pre-benchmark audit information and inserts a timestamp into a database audit 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 controlling 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.

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 effective 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 availability 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:	**9198.37 tpmC**
PRICE per tpmC:	**$50.35 per tpmC**
HARDWARE AVAILABILITY:	**current**
SOFTWARE AVAILABILITY:	**11/30/97**

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 sponsor 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: Jim Nagler
10450 Ridgeview Court
PO Box 4050, MS 49EL-FS
Cupertino, CA 95015-4050
phone: 408 343-6332

PERFORMANCE METRICS INC.
TPC Certified Auditors

Anh Nguyen and Jim Nagler
Network Server Division
Hewlett-Packard Company
5301 Stevens Creek Boulevard
Santa Clara, CA 95052-8059

July 7, 1997

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

Platform: Hewlett-Packard NetServer LX Pro
Database Manager: Microsoft SQL Server 6.5 Enterprise Edition
Operating System: Microsoft Windows NT Server 4.0 Enterprise Edition
Transaction Manager: Tuxedo CFS 6.3

CPUs	Memory	Disks	New-Order Response Time @ 90%	tpmC
Server: HP NetServer LX Pro				
4 Pentium Pro @ 200 MHz	Main: 4 GB L2 Cache: 512 KB	40 @ 3.97 GB 63 @ 8.47 GB	1.69 sec.	9,198.37
4 Clients: HP NetServer E40				
1 Pentium Pro @ 200 MHz	Main: 256 MB L2 Cache: 256 KB	1 @ 1.98 GB	n.a.	n.a.

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

The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 820 warehouses. Only 785 warehouses were active during measurement.
- The ACID properties were met, including phantom protection.
- The durability data loss test was performed on a 10-warehouse database.
- Input data was generated according to the specified percentages.

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

- It was demonstrated that the following disks on the server were not used during measurement: 4 at 3.97 GB and 2 at 8.47 GB.
- Eight hours of mirrored log space was configured on the priced system.
- The data for the 180-day space calculation was verified. The measured system contained sufficient storage to satisfy this requirement.
- Measurement cycle times included a 0.1 second menu and a 0.1 second response time delay for an emulated Web browser.
- The steady state portion of the test was 30 minutes.
- One checkpoint was taken during the steady state portion of the test.
- Checkpoints were verified to be clear of the guard zones.
- There were 7,850 user contexts present on the system.
- Each emulated user had a unique starting random number seed.
- The NURand constants used for database load and at run time were verified.
- System pricing was checked for major components and maintenance.

Additional Audit Notes: none

Regards,

Richard L. Gimarc
Auditor

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

Appendix A – 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.

db.h

```
    #ifdef USE_ODBC
        void dbsetuserdata(PDBPROCESS dbproc, void *uPtr);
        void *dbgetuserdata(PDBPROCESS dbproc);
        void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD fCType,
    SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue, SDWORD
    cbValueMax);
        void ODBCError(PDBPROCESS dbproc);
        BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement);
        BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol, SQLSMALLINT
    fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax);
        BOOL GetResults(PDBPROCESS dbproc);
        BOOL MoreResults(PDBPROCESS  dbproc);
        BOOL ReopenConnection(PDBPROCESS dbproc);
    #endif
```

dbtune2

```
    Key Name:           SOFTWARE\Microsoft\MSSQLServer
    Class Name:         <NO CLASS>
    Last Write Time:    5/8/97 - 6:09 PM

    Key Name:           SOFTWARE\Microsoft\MSSQLServer\Client
    Class Name:         <NO CLASS>
    Last Write Time:    5/8/97 - 6:13 PM

    Key Name:           SOFTWARE\Microsoft\MSSQLServer\Client\DB-Lib
    Class Name:         <NO CLASS>
    Last Write Time:    6/12/97 - 9:03 PM
    Value 0
      Name:             AutoAnsiToOem
      Type:             REG_SZ
      Data:             on

Value 1
  Name:             UseIntlSettings
  Type:             REG_SZ
  Data:             ON


Key Name:           SOFTWARE\Microsoft\MSSQLServer\MSSQLServer
Class Name:         <NO CLASS>
Last Write Time:    6/4/97 - 1:59 PM
Value 0
  Name:             AuditLevel
  Type:             REG_DWORD
  Data:             0

Value 1
  Name:             DefaultDomain
  Type:             REG_SZ
  Data:             NLDSUT1

Value 2
  Name:             DefaultLogin
  Type:             REG_SZ
  Data:             guest

Value 3
  Name:             ImpersonateClient
  Type:             REG_DWORD
  Data:             0

Value 4
  Name:             ListenOn
  Type:             REG_MULTI_SZ
  Data:             SSNMPN60,\\.\pipe\sql\query
                    SSMSSO60,1433


Value 5
  Name:             LoginMode
  Type:             REG_DWORD
  Data:             0

Value 6
  Name:             Map#
  Type:             REG_SZ
  Data:             -
```

```
Value 7
  Name:            Map$
  Type:            REG_SZ
  Data:

Value 8
  Name:            Map_
  Type:            REG_SZ
  Data:            \

Value 9
  Name:            ResourceMgrID
  Type:            REG_SZ
  Data:            {7FF533C2-C809-11D0-B4B0-0060B03C95CF}

Value 10
  Name:            SetHostname
  Type:            REG_DWORD
  Data:            0

Value 11
  Name:            Tapeloadwaittime
  Type:            REG_DWORD
  Data:            0xffffffff


Key Name:
SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\CurrentVersion
Class Name:        <NO CLASS>
Last Write Time:   5/8/97 - 7:30 PM
Value 0
  Name:            CurrentVersion
  Type:            REG_SZ
  Data:            6.50.252

Value 1
  Name:            RegisteredOrganization
  Type:            REG_SZ
  Data:            Hewlett Packard

Value 2
  Name:            RegisteredOwner
  Type:            REG_SZ
  Data:            Jim Nagler

Value 3
  Name:            RegisteredProductID
  Type:            REG_SZ
  Data:

Value 4
  Name:            SerialNumber
  Type:            REG_DWORD
  Data:            0x83980040

Value 5
  Name:            SoftwareType
  Type:            REG_SZ
  Data:            System


Key Name:          SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\Parameters
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 1:59 PM
Value 0
  Name:            SQLArg0
  Type:            REG_SZ
  Data:            -dC:\MSSQL\DATA\MASTER.DAT

Value 1
  Name:            SQLArg1
  Type:            REG_SZ
  Data:            -eC:\MSSQL\LOG\ERRORLOG


Key Name:          SOFTWARE\Microsoft\MSSQLServer\Replication
Class Name:        <NO CLASS>
Last Write Time:   5/8/97 - 6:28 PM
Value 0
  Name:            DistributionDB
  Type:            REG_SZ
  Data:

Value 1
  Name:            WorkingDirectory
  Type:            REG_SZ
  Data:            C:\MSSQL\REPLDATA


Key Name:          SOFTWARE\Microsoft\MSSQLServer\Setup
Class Name:        <NO CLASS>
Last Write Time:   6/4/97 - 1:59 PM
Value 0
  Name:            CRC
```

```
  Type:            REG_SZ
  Data:            130876680

Value 1
  Name:            SetupStatus
  Type:            REG_SZ
  Data:            Installed

Value 2
  Name:            SQLPath
  Type:            REG_SZ
  Data:            C:\MSSQL


Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQL Interface
Class Name:        REG_MULTI_SZ
Last Write Time:   6/13/97 - 5:45 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQL Interface\Graph
Control
Class Name:        REG_MULTI_SZ
Last Write Time:   6/13/97 - 5:45 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQL Service Manager
Class Name:        <NO CLASS>
Last Write Time:   5/8/97 - 6:28 PM
Value 0
  Name:            Action Verify
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            DefaultSvc
  Type:            REG_SZ
  Data:            MSSQLServer

Value 2
  Name:            Remote
  Type:            REG_DWORD
  Data:            0x1

Value 3
  Name:            Services
  Type:            REG_MULTI_SZ
  Data:            MSSQLServer
                   SQLExecutive
                   MSDTC

Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQLExecutive
Class Name:        <NO CLASS>
Last Write Time:   5/8/97 - 6:28 PM
Value 0
  Name:            CmdExecAccount
  Type:            REG_BINARY
  Data:
00000000   41 ac 8a 68 cf de 87 4e - a4 42 be 7c d4 ec 5f 15
A..h...N.B.|.._.

Value 1
  Name:            MailAutoStart
  Type:            REG_DWORD
  Data:            0x1

Value 2
  Name:            NonAlertableErrors
  Type:            REG_SZ
  Data:            1204,4002

Value 3
  Name:            RestartSQLServer
  Type:            REG_DWORD
  Data:            0x1

Value 4
  Name:            RestartSQLServerInterval
  Type:            REG_DWORD
  Data:            0x5

Value 5
  Name:            ServerHost
  Type:            REG_SZ
  Data:

Value 6
  Name:            SyshistoryLimitRows
  Type:            REG_DWORD
  Data:            0x1

Value 7
  Name:            SyshistoryMaxRows
  Type:            REG_DWORD
  Data:            0x3e8
```

```
Value 8
  Name:            TaskHistoryMaxRows
  Type:            REG_DWORD
  Data:            0x64


Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQLExecutive\Subsystems
Class Name:        <NO CLASS>
Last Write Time:   5/8/97 - 6:28 PM
Value 0
  Name:            CmdExec
  Type:            REG_SZ
  Data:
C:\MSSQL\BINN\CMDEXEC.DLL,CmdExecStart,CmdEvent,CmdExecStop,10

Value 1
  Name:            Distribution
  Type:            REG_SZ
  Data:
C:\MSSQL\BINN\SQLREPL.DLL,distribution_start,distribution_event,distribut
ion_stop,100

Value 2
  Name:            LogReader
  Type:            REG_SZ
  Data:
C:\MSSQL\BINN\SQLREPL.DLL,logreader_start,logreader_event,logreader_stop,
25

Value 3
  Name:            Sync
  Type:            REG_SZ
  Data:
C:\MSSQL\BINN\SQLREPL.DLL,sync_start,sync_event,sync_stop,100
```

DELIRPT.c

```
/*  FILE:DELIRPT.C
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:Delivery report processing application
 *  Author:Philip Durr
 *  philipdu@Microsoft.com
 */
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#define LOGFILE_READ_EOF0//check log file flag return current state
#define LOGFILE_CLEAR_EOF1//clear end of log file flag
#define LOGFILE_SET_EOF2//set flag end of log file reached
#define INTERVAL.01//90th percentile calculation bucket interval
#define ERR_SUCCESS1000//success no error
#define ERR_READING_LOGFILE1001//io errors occured reading delivery log
file
#define ERR_INSUFFICIENT_MEMORY1002//insuficient memory to process 90th
percentile report
#define ERR_CANNOT_OPEN_RESULTS_FILE1005//Cannot open delivery results
file delilog.
typedef struct _RPTLINE
{
    SYSTEMTIMEstart;//delilog report line start time
    SYSTEMTIMEend;//delilog report line end time
    intresponse;//delilog report line time delivery took in milliseconds
    intw_id;//delilog report line warehouse id for delivery
    into_carrier_id;//delilog report line carier id for delivery
    intitems[10];//delilog report line delivery line items
} RPTLINE, *PRPTLINE;
//error message structure used in ErrorMessage API
typedef struct _SERRORMSG
{
    intiError;//error id of message
    charszMsg[80];//message to sent to browser
} SERRORMSG;
int versionMS = 4;//delirpt version
int versionMM = 0;
int versionLS = 0;
int iReport;//delirpt report to process
int iStartTime;//begin times to accept for report
int iEndTime;//end times to accept for report
FILE*fpLog;//log file stream
//Local function prototypes
voidmain(int argc, char *argv[]);
static intInit(void);
static voidRestore(void);
static intDoReport(void);
int AverageResponse(void);
int SkippedDelivery(void);
int Percentile90th(void);
BOOLCheckTimes(PRPTLINE pRptLine);
```

```
static intOpenLogFile(void);
static voidCloseLogFile(void);
static voidResetLogFile(void);
static BOOLLogEOF(int iOperation);
static BOOLReadReportLine(char *szBuffer, PRPTLINE pRptLine);
static BOOLParseReportLine(char *szLine, PRPTLINE pRptLine);
static BOOLParseDate(char *szDate, LPSYSTEMTIME pTime);
static BOOLParseTime(char *szTime, LPSYSTEMTIME pTime);
static voidErrorMessage(int iError);
static BOOLGetParameters(int argc, char *argv[]);
static voidPrintParameters(void);
static voidPrintHeader(void);
static voidcls(void);
static BOOLIsNumeric(char *ptr);
/* FUNCTION: int main(int argc, char *argv[])
 *
 * PURPOSE:This function is the beginning execution point for the
delivery executable.
 *
 * ARGUMENTS:intargcnumber of command line arguments passed to delivery
 *  char*argv[]array of command line argument pointers
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void main(int argc, char *argv[])
{
    intiError;
    PrintHeader();
    if ( GetParameters(argc, argv) )
    {
    PrintParameters();
    return;
    }
    if ( (iError=Init()) != ERR_SUCCESS )
    {
    ErrorMessage(iError);
    Restore();
    return;
    }
    if ( (iError = DoReport()) != ERR_SUCCESS )
    ErrorMessage(iError);
    Restore();
    return;
}
/* FUNCTION: static int Init(void)
 *
 * PURPOSE:This function initializes the delirtp application.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static int Init(void)
{
    int iError;
    if ( (iError = OpenLogFile()) )
    return iError;
    return TRUE;
}
/* FUNCTION: static void Restore(void)
 *
 * PURPOSE:This function cleans up the delirpt application before
termination.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void Restore(void)
{
    CloseLogFile();
    return;
}
/* FUNCTION: static int DoReport(void)
 *
 * PURPOSE:This function dispatches the requested report.
 *
 * ARGUMENTS:None
 *
 * RETURNS:ERR_SUCCESS if successfull or error code if an error occurs.
 *
 * COMMENTS:None
 *
 */
static int DoReport(void)
```

```
{
    int iRc;
    switch(iReport)
    {
    case 1:
    iRc = AverageResponse();
    break;
    case 2:
    iRc = Percentile90th();
    break;
    case 3:
    iRc = SkippedDelivery();
    break;
    case 4:
    if ( (iRc = AverageResponse()) != ERR_SUCCESS )
    break;
    if ( (iRc = Percentile90th()) != ERR_SUCCESS )
    break;
    if ( (iRc = SkippedDelivery()) != ERR_SUCCESS )
    break;
    break;
    }
    return iRc;
}
/* FUNCTION: int AverageResponse(void)
 *
 * PURPOSE:This function processes the AverageResponse report.
 *
 * ARGUMENTS:None
 *
 * RETURNS:ERR_SUCCESS if successfull or error code if an error occurs.
 *
 * COMMENTS:None
 *
 */
int AverageResponse(void)
{
    RPTLINE reportLine;
    intiTotalResponse;
    intiLines;
    doublefAverage;
    charszDelivery[128];
    ResetLogFile();
    iTotalResponse = 0;
    iLines = 0;
    printf("\n\n******** Average Response Time Report *******\n");
    while ( !LogEOF(LOGFILE_READ_EOF) )
    {
    if ( ReadReportLine(szDelivery, &reportLine) )
    return ERR_READING_LOGFILE;
    if ( !LogEOF(LOGFILE_READ_EOF) )
    {
    if ( CheckTimes(&reportLine) )
    continue;
    iLines++;
    iTotalResponse += reportLine.response;
    if ( iLines % 10 == 0 )
    printf("Reading Report Line:\t%d\r", iLines);
    }
    }
    printf("                                          \r");
    if ( iLines == 0 )
    {
    printf("No deliveries found.\n");
    }
    else
    {
    fAverage = ((double)iTotalResponse / (double)iLines)/(double)1000;
    printf("Total Deliveries:      %10.0f\n", (float)iLines);
    printf("Total Response Times:  %10.3f\n",
((float)iTotalResponse/(float)1000));
    printf("Average Response Time: %10.3f\n", fAverage);
    }
    return ERR_SUCCESS;
}
/* FUNCTION: int Percentile90th(void)
 *
 * PURPOSE:This function processes the 90th percentile report.
 *
 * ARGUMENTS:None
 *
 * RETURNS:ERR_SUCCESS if successfull or error code if an error occurs.
 *
 * COMMENTS:This function requires enough space to allocate needed
 *  buckets which will be 2 * max response time in
 *  deci-seconds.
 *
 */
int Percentile90th(void)
{
    RPTLINE reportLine;
    intiBucketSize;
    inti;
    intiResponseSeconds;
```

```
intiMaxSeconds;
intiTotalBuckets;
doubleiTotal;
doublei90thPercent;
short*psBuckets;
charszDelivery[128];

printf("\n\n******** 90th Percentile *******\n");
printf("Calculating Max Response Seconds...\n");
ResetLogFile();
iMaxSeconds = -1;
while ( !LogEOF(LOGFILE_READ_EOF) )
{
if ( ReadReportLine(szDelivery, &reportLine) )
return ERR_READING_LOGFILE;
if ( szDelivery[0] == '*' )
continue;
if ( !LogEOF(LOGFILE_READ_EOF) )
{
if ( iMaxSeconds < reportLine.response )
iMaxSeconds = reportLine.response;
}
}
iTotalBuckets = iMaxSeconds + 1;
printf("Allocating Buckets...\n");
iBucketSize = iTotalBuckets * sizeof(short);
if ( !(psBuckets = (short *)malloc(iBucketSize)) )
return ERR_INSUFFICIENT_MEMORY;
ZeroMemory(psBuckets, iBucketSize);
iTotal = 0;
ResetLogFile();
printf("Calculating Distribution...\n");
while ( !LogEOF(LOGFILE_READ_EOF) )
{
if ( ReadReportLine(szDelivery, &reportLine) )
return ERR_READING_LOGFILE;
if ( szDelivery[0] == '*' )
continue;
if ( !LogEOF(LOGFILE_READ_EOF) )
{
if ( CheckTimes(&reportLine) )
continue;
psBuckets[reportLine.response]++;
iTotal++;
}
}
i90thPercent = iTotal * .9;
    for(i=0, iTotal = 0.0; iTotal < i90thPercent; iTotal +=
(double)psBuckets[i] )
    i++;
    printf("90th Percentile = %d.%d\n", i/1000, (i % 1000));
    free(psBuckets);
    return ERR_SUCCESS;
}
/* FUNCTION: int SkippedDelivery(void)
 *
 * PURPOSE:This function processes the Skipped Deliveries report.
 *
 * ARGUMENTS:None
 *
 * RETURNS:ERR_SUCCESS if successfull or error code if an error occurs.
 *
 * COMMENTS:None
 *
 */
int SkippedDelivery(void)
{
    RPTLINE reportLine;
    charszDelivery[128];
    inti;
    intitems[10];
    ResetLogFile();
    printf("\n\n******** Skipped Delivery Report *******\n");
    memset(items, 0, sizeof(items));
    printf("Reading Delivery Log File...");
    while ( !LogEOF(LOGFILE_READ_EOF) )
    {
    if ( ReadReportLine(szDelivery, &reportLine) )
    return ERR_READING_LOGFILE;
    if ( !LogEOF(LOGFILE_READ_EOF) )
    {
    if ( CheckTimes(&reportLine) )
    continue;
    for(i=0; i<10; i++)
    {
    if ( !reportLine.items[i] )
    items[i]++;
    }
    }
    }
    printf("\n");
    printf("Skipped delivery table.\n");
    printf("  1    2    3    4    5    6    7    8    9   10 \n");
    printf("---- ---- ---- ---- ---- ---- ---- ---- ---- ----\n");
```

```
    for(i=0; i<10; i++)
    printf("%4.4d ", items[i]);
    printf("\n");
    return ERR_SUCCESS;
}
/* FUNCTION: BOOL CheckTimes(PRPTLINE pRptLine)
 *
 * PURPOSE:This function checks to see of the delilog record falls
withing the
 *  begin and end time from the command line.
 *
 * ARGUMENTS:PRPTLINEpRptLinedelilog processed report line.
 *
 * RETURNS:BOOLFALSEif report line is not within the
 *  requested start and end times.
 *  TRUEif the report line is within the
 *  requested start and end times.
 *
 * COMMENTS:If startTime and endTime are both 0 then the user requested
 *  the default behavior which is all records in delilog are
 *  valid.
 */
BOOL CheckTimes(PRPTLINE pRptLine)
{
    intiRptEndTime;
    intiRptStartTime;
    iRptStartTime = (pRptLine->start.wHour * 3600000) + (pRptLine-
>start.wMinute * 60000) + (pRptLine->start.wSecond * 1000) + pRptLine-
>start.wMilliseconds;
    iRptEndTime = (pRptLine->end.wHour * 3600000) + (pRptLine-
>end.wMinute * 60000) + (pRptLine->end.wSecond * 1000) + pRptLine-
>end.wMilliseconds;
    if ( iStartTime == 0 && iEndTime == 0 )
    return FALSE;
    if ( iStartTime <= iRptStartTime && iEndTime >= iRptEndTime )
    return FALSE;
    return TRUE;
}
/* FUNCTION: int OpenLogFile(void)
 *
 * PURPOSE:This function opens the delivery log file for use.
 *
 * ARGUMENTS:None
 *
 * RETURNS:intERR_CANNOT_OPEN_RESULTS_FILECannot create results log file.
 *  ERR_SUCCESSLog file successfully opened
 *
 *
 * COMMENTS:None
 *
 */
static int OpenLogFile(void)
{
    fpLog = fopen("delilog.", "rb");
    if ( !fpLog )
    return ERR_CANNOT_OPEN_RESULTS_FILE;
    return ERR_SUCCESS;
}
/* FUNCTION: int CloseLogFile(void)
 *
 * PURPOSE:This function closes the delivery log file.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void CloseLogFile(void)
{
    if ( fpLog )
    fclose(fpLog);
    return;
}
/* FUNCTION: static void ResetLogFile(void)
 *
 * PURPOSE:This function prepares the delilog. file for reading
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ResetLogFile(void)
{
    fseek(fpLog, 0L, SEEK_SET);
    LogEOF(LOGFILE_CLEAR_EOF);
    return;
}
/* FUNCTION: static BOOL LogEOF(int iOperation)
 *
 * PURPOSE:This function tracks and reports the end of file condition
```

```
 *  on the delilog file.
 *
 * ARGUMENTS:int iOperationrequested operation this can be:
 *  LOGFILE_READ_EOFcheck log file flag return current state
 *  LOGFILE_CLEAR_EOFclear end of log file flag
 *  LOGFILE_SET_EOFset flag end of log file reached
 *
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static BOOL LogEOF(int iOperation)
{
    static BOOL bEOF;
    switch(iOperation)
    {
    case LOGFILE_READ_EOF:
    return bEOF;
    break;
    case LOGFILE_CLEAR_EOF:
    bEOF = FALSE;
    break;
    case LOGFILE_SET_EOF:
    bEOF = TRUE;
    break;
    }
    return FALSE;
}
/* FUNCTION: static BOOL ReadReportLine(char *szBuffer, PRPTLINE
pRptLine)
 *
 * PURPOSE:This function reads a text line from the delilog file.
 *  on the delilog file.
 *
 * ARGUMENTS:char*szBufferbuffer to placed read delilog file line into.
 *  PRPTLINEpRptLinereturned structure containing parsed delilog
 *  report line.
 *
 * RETURNS:FALSEif 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*szLinebuffer containing the delilog file line to be
parsed.
 *  PRPTLINEpRptLinereturned structure containing parsed delilog
 *  report line values.
 *
 * RETURNS:FALSEif 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*szDatebuffer containing the date to be parsed.
 *  LPSYSTEMTIMEpTimesystem 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*szTimebuffer containing the time to be parsed.
 *  LPSYSTEMTIMEpTimesystem time structure where date will be placed.
 *
 * RETURNS:FALSE if successfull or TRUE if an error occurs.
 *
 * COMMENTS:None
 *
 */
static BOOL ParseTime(char *szTime, LPSYSTEMTIME pTime)
```

```
{
    if ( !isdigit(*szTime) || !isdigit(*(szTime+1)) || *(szTime+2) !=
':' ||
    !isdigit(*(szTime+3)) || !isdigit(*(szTime+4)) || *(szTime+5) != ':'
||
     !isdigit(*(szTime+6)) || !isdigit(*(szTime+7)) || *(szTime+8) !=
':' ||
     !isdigit(*(szTime+9)) || !isdigit(*(szTime+10)) ||
!isdigit(*(szTime+11)) )
     return TRUE;
    pTime->wHour = atoi(szTime);
    pTime->wMinute = atoi(szTime+3);
    pTime->wSecond = atoi(szTime+6);
    pTime->wMilliseconds  = atoi(szTime+9);
    if ( pTime->wHour > 23 || pTime->wHour < 0 ||
    pTime->wMinute > 59 || pTime->wMinute < 0 ||
    pTime->wSecond > 59 || pTime->wSecond < 0 ||
    pTime->wMilliseconds < 0 )
    return TRUE;
    if ( pTime->wMilliseconds > 999 )
    {
    pTime->wSecond += (pTime->wMilliseconds/1000);
    pTime->wMilliseconds = pTime->wMilliseconds  % 1000;
    }
    return FALSE;
}
/* FUNCTION: void ErrorMessage(int iError)
 *
 * PURPOSE:This function displays an error message in the delivery
executable's console window.
 *
 * ARGUMENTS:intiErrorerror id to be displayed
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ErrorMessage(int iError)
{
    int i;
    static SERRORMSG errorMsgs[] =
    {
    {ERR_SUCCESS,"Success, no error."},
    {ERR_CANNOT_OPEN_RESULTS_FILE,"Cannot open delivery results file
delilog."},
    {ERR_READING_LOGFILE,"Reading delivery log file, Delivery item
format incorrect."},
    {ERR_INSUFFICIENT_MEMORY,"insufficient memory to process 90th
percentile report."},
    {0,""}
    };
    for(i=0; errorMsgs[i].szMsg[0]; i++)
    {
    if ( iError == errorMsgs[i].iError )
    {
    printf("\nError(%d): %s", iError, errorMsgs[i].szMsg);
    return;
    }
    }
    printf("Error(%d): %s", errorMsgs[0].szMsg);
    return;
}
/* FUNCTION: BOOL GetParameters(int argc, char *argv[])
 *
 * PURPOSE:This function parses the command line passed in to the
delivery executable, initializing
 *  and filling in global variable parameters.
 *
 * ARGUMENTS:intargcnumber of command line arguments passed to delivery
 *  char*argv[]array of command line argument pointers
 *
 * RETURNS:BOOLFALSEparameter read successfull
 *  TRUEuser has requested parameter information screen be displayed.
 *
 * COMMENTS:None
 *
 */
static BOOL GetParameters(int argc, char *argv[])
{
    inti;
    SYSTEMTIMEstartTime;
    SYSTEMTIMEendTime;
    iStartTime = 0;
    iEndTime = 0;
    iReport = 4;
    for(i=0; i<argc; i++)
    {
    if ( argv[i][0] == '-' || argv[i][0] == '/' )
    {
    switch(argv[i][1])
    {
    case 'S':
    case 's':
    if ( ParseTime(argv[i]+2, &startTime) )
```

```
    return TRUE;
    iStartTime = (startTime.wHour * 3600000) + (startTime.wMinute *
60000) + (startTime.wSecond * 1000) + startTime.wMilliseconds;
    break;
    case 'E':
    case 'e':
    if ( ParseTime(argv[i]+2, &endTime) )
    return TRUE;
    iEndTime = (endTime.wHour * 3600000) + (endTime.wMinute * 60000) +
(endTime.wSecond * 1000) + endTime.wMilliseconds;
    break;
    case 'R':
    case 'r':
    iReport = atoi(argv[i]+2);
    if ( iReport > 4 || iReport < 1 )
    iReport = 4;
    break;
    case '?':
    return TRUE;
    }
    }
    }
    return FALSE;
}
/* FUNCTION: void PrintParameters(void)
 *
 * PURPOSE:This function displays the supported command line flags.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void PrintParameters(void)
{
    PrintHeader();
    printf("DELIRPT:\n\n");

printf("Parameter
Default\n");
    printf("-----------------------------------------------------------------
----------\n");
    printf("-S Start Time
HH:MM:SS:MMM                                    All    \n");
    printf("-E End Time
HH:MM:SS:MMM                                      All    \n");
    printf("-R 1)Average Response, 2)90th 3) Skipped 4)
All                 All    \n");
    printf("-? This help screen\n\n");
    printf("Note:  Command line switches are NOT case sensitive.\n");
    return;
}
/* FUNCTION: void PrintHeader(void)
 *
 * PURPOSE:This function displays the delivery report applications
banner information.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void PrintHeader(void)
{
    cls();
    printf("************************************************\n");
    printf("*                                              *\n");
    printf("*  Microsoft SQL Server 6.5                    *\n");
    printf("*                                              *\n");
    printf("*  HTML TPC-C BENCHMARK KIT: Delivery Report    *\n");
    printf("*  Version %d.%2.2d.%3.3d                           *\n",
versionMS, versionMM, versionLS);
    printf("*                                              *\n");
    printf("************************************************\n\n");
    return;
}
/* FUNCTION: void cls(void)
 *
 * PURPOSE:This function clears the console window
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void cls(void)
{
    HANDLEhConsole;
    COORDcoordScreen = { 0, 0 };//here's where we'll home the cursor
```

```
    DWORDcCharsWritten;
    CONSOLE_SCREEN_BUFFER_INFOcsbi;//to get buffer info
    DWORDdwConSize;//number of character cells in the current buffer
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    //get the number of character cells in the current buffer
    GetConsoleScreenBufferInfo( hConsole, &csbi );
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
    //fill the entire screen with blanks
    FillConsoleOutputCharacter( hConsole, (TCHAR) ' ', dwConSize,
coordScreen, &cCharsWritten );
    GetConsoleScreenBufferInfo( hConsole, &csbi );

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

    //put the cursor at (0, 0)
    SetConsoleCursorPosition( hConsole, coordScreen );
    return;
}
/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE:This function determines if a string is numeric. It fails if
any characters other
 *  than numeric and null terminator are present.
 *
 * ARGUMENTS:char*ptrpointer to string to check.
 *
 * RETURNS:BOOLFALSEif string is not all numeric
 *  TRUEif string contains only numeric characters i.e. '0' - '9'
 *
 * COMMENTS:A comma is counted as a valid delimiter.
 *
 */
static BOOL IsNumeric(char *ptr)
{
    if ( *ptr == 0 )
    return FALSE;
    while( *ptr && isdigit(*ptr) )
    ptr++;
    if ( !*ptr || *ptr == ',' )
    return TRUE;
    else
    return FALSE;
}
```

Delisrc.c

```
/*  FILE:DELISRV.C
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96, By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:Delivery TPC-C transaction executable
 *  Author:Philip Durr
 *  philipdu@Microsoft.com
 */
#include <windows.h>
#include <process.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include  <io.h>
#include <conio.h>
#include <ctype.h>
#ifdef USE_ODBC
    #include <sql.h>
    #include <sqlext.h>
    HENVhenv;
#else
    #define DBNTWIN32
    #include <sqlfront.h>
    #include <sqldb.h>
#endif
#include "delisrv.h"
charszServer[32];//SQL server name
charszDatabase[32];//tpcc database name
charszUser[32];//user name
charszPassword[32];//user password
int iNumThreads= 4;//number of threads to create
int iDelayMs= 1000;//delay between delivery queue checks
int iDeadlockRetry= 3;//number of read check retries.
int iQSlotts= 3000;//delivery transaction queues
int iConnectDelay= 500;//delay between re-connect attempts if sql server
refuses connection.
FILE*fpLog;//pointer to log file
CRITICAL_SECTIONWriteLogCriticalSection;//critical section for delivery
write log
```

```
CRITICAL_SECTIONDeliveryCriticalSection;//critical section for delivery
transactions cache
staticLPTSTRlpszPipeName = TEXT("\\\\.\\pipe\\DELISRV");//delivery pipe
name
HANDLEhPipe= INVALID_HANDLE_VALUE;//delivery pipe handle
HANDLEhComPort= INVALID_HANDLE_VALUE;//delivery pipe completion port
handle.
BOOLbDone;//delivery executable termination request flag
BOOLbFlush;//Flush delivery log info when written.
LPDELIVERY_PACKETpDeliveryCache;
int versionMS = 4;//delivery executable version number.
int versionMM = 0;//formatted as MS.MM.LS, 1.00.005
int versionLS = 0;
/* FUNCTION: int main(int argc, char *argv[])
 *
 * PURPOSE:This function is the beginning execution point for the
delivery executable.
 *
 * ARGUMENTS:intargcnumber of command line arguments passed to delivery
 *  char*argv[]array of command line argument pointers
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void main(int argc, char *argv[])
{
    intiError;
    if ( GetParameters(argc, argv) )
    {
    PrintParameters();
    return;
    }
    if ( (iError=Init()) )
    {
    ErrorMessage(iError);
    Restore();
    return;
    }
    if ( (iError = RunDelivery()) != ERR_SUCCESS )
    ErrorMessage(iError);
    Restore();
    return;
}
/* FUNCTION: void cls(void)
 *
 * PURPOSE:This function clears the console window
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void cls(void)
{
    HANDLEhConsole;
    COORDcoordScreen = { 0, 0 };//here's where we'll home the cursor
    DWORDcCharsWritten;
    CONSOLE_SCREEN_BUFFER_INFOcsbi;//to get buffer info
    DWORDdwConSize;//number of character cells in the current buffer
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    //get the number of character cells in the current buffer
    GetConsoleScreenBufferInfo( hConsole, &csbi );
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;
    //fill the entire screen with blanks
    FillConsoleOutputCharacter( hConsole, (TCHAR) ' ', dwConSize,
coordScreen, &cCharsWritten );
    GetConsoleScreenBufferInfo( hConsole, &csbi );

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

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

/* FUNCTION: int RunDelivery(void)
 *
 * PURPOSE:This function executes the main delivery executable loop.
 *
 * ARGUMENTS:None
 *
 * RETURNS:intERR_CANNOT_OPEN_PIPEcannot open named pipe
 *  ERR_CANNOT_CREATE_THREADcannot create required threads
 *  ERR_SUCCESSsuccessfull no error
 *
 *
 * COMMENTS:None
 *
```

```
 */
static int RunDelivery(void)
{
    SECURITY_ATTRIBUTESsa;
    inti;
    cls();
    PrintHeader();
    printf("\n<Starting Delivery Service with %d Threads.>\n",
iNumThreads);
    printf("\nPress <Ctrl>C to exit.\n");
    bDone = FALSE;
    _beginthread( CheckKey, 0, NULL );
    printf("\nWaiting for delivery pipe: ");
    while( !bDone )
    {
    AnimateWait1();
    if ( WaitNamedPipe(lpszPipeName, NMPWAIT_USE_DEFAULT_WAIT) )
    {
    sa.nLength= sizeof(sa);
    sa.lpSecurityDescriptor= NULL;
    sa.bInheritHandle= TRUE;
    hPipe = CreateFile(lpszPipeName, GENERIC_READ | GENERIC_WRITE,
FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING,
    FILE_FLAG_OVERLAPPED, NULL);
    if ( hPipe  == INVALID_HANDLE_VALUE )
    return ERR_CANNOT_OPEN_PIPE;
    hComPort = CreateIoCompletionPort(hPipe, NULL, 0, 256);
    break;
    }
    Sleep(100);
    }
    if ( !bDone )
    {
    if ( _beginthread( DeliveryHandler, 0, NULL ) == -1 )
    return ERR_CANNOT_CREATE_THREAD;
    for(i=0; i<iNumThreads; i++)
    {
    if ( _beginthread( DeliveryThread, 0, NULL ) == -1 )
    return ERR_CANNOT_CREATE_THREAD;
    }
    printf(" \nRunning : ");
    while( !bDone )
    AnimateWait();
    }
    return ERR_SUCCESS;
}
/* FUNCTION: void AnimateWait1(void)
 *
 * PURPOSE:This function provides a visual indicator that the delivery
executable is waiting for
 *  the delivery pipe to appear.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void AnimateWait1(void)
{
    const static char szStr[] = "+-|*";
    static char *ptr = (char *)szStr;
    printf("%c\x8", *ptr);
    ptr = (*(ptr+1)) ? ptr + 1 : (char *)szStr;
    Sleep(100);
    return;
}
/* FUNCTION: void AnimateWait(void)
 *
 * PURPOSE:This function provides a visual indicator that the delivery
executable is waiting for
 *  and processing transactions.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void AnimateWait(void)
{
    const static char szStr[] = "/-\\|/-\\|";
    static char *ptr = (char *)szStr;
    printf("%c\x8", *ptr);
    ptr = (*(ptr+1)) ? ptr + 1 : (char *)szStr;
    Sleep(100);
    return;
}
/* FUNCTION: int Init(void)
 *
 * PURPOSE:This function prepares the delivery executable for processing.
 *
```

```
 * ARGUMENTS:None
 *
 * RETURNS:intiErrorError code if unsuccessfull
 *  ERR_SUCCESSNo error successfull code
 *
 *
 * COMMENTS:None
 *
 */
static int Init(void)
{
    intiError;
    InitializeCriticalSection(&WriteLogCriticalSection);
    InitializeCriticalSection(&DeliveryCriticalSection);
    fpLog= NULL;
    if ( !(pDeliveryCache = malloc(sizeof(DELIVERY_PACKET) * iQSlotts)) )
    return ERR_INSUFFICIENT_MEMORY;
    memset(pDeliveryCache, 0, sizeof(DELIVERY_PACKET) * iQSlotts);
    if ( (iError = ReadRegistrySettings()) )
    return iError;
    if ( (iError=OpenLogFile()) )
    return iError;
    //initialize db library for use
#ifdef USE_ODBC
    if ( SQLAllocEnv(&henv) == SQL_ERROR )
    return ERR_ODBC_SQLALLOCENV;
#else
    dbinit();
    // install Db Library error and message handlers
    dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
    dberrhandle((DBERRHANDLE_PROC)err_handler);
#endif
    return ERR_SUCCESS;
}
/* FUNCTION: void Restore(void)
 *
 * PURPOSE:This function cleans up allocated objects to allow for
termination of the
 *  delivery executable.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void Restore(void)
{
    intiret, l, d;
    DeleteCriticalSection(&WriteLogCriticalSection);
    DeleteCriticalSection(&DeliveryCriticalSection);
    l = 1;
    iret = WriteFile(hPipe, &l, 1, &d, NULL);
    if ( hPipe  != INVALID_HANDLE_VALUE )
    iret = CloseHandle(hPipe);
    if ( fpLog )
    fclose(fpLog);
    fpLog = NULL;
#ifdef USE_ODBC
    SQLFreeEnv(henv);
#else
    dbexit();
#endif
    return;
}
/* FUNCTION: void ErrorMessage(int iError)
 *
 * PURPOSE:This function displays an error message in the delivery
executable's console window.
 *
 * ARGUMENTS:intiErrorerror id to be displayed
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ErrorMessage(int iError)
{
    int i;
    static SERRORMSG errorMsgs[] =
    {
    {ERR_SUCCESS,"Success, no error."},
    {ERR_CANNOT_CREATE_THREAD,"Cannot create thread."},
    {ERR_DBGETDATA_FAILED,"Get data failed."},
    {ERR_REGISTRY_NOT_SETUP,"Registry not setup for tpcc."},
    {ERR_CANNOT_ACCESS_DELIVERY_FN,"Cannot access ReadDelivery cache."},
    {ERR_CANNOT_ACCESS_REGISTRY,"Cannot access registry key TPCC."},
    {ERR_CANNOT_CREATE_RESULTS_FILE,"Cannot create results file."},
    {ERR_CANNOT_OPEN_PIPE,"Cannot open delivery pipe."},
    {ERR_READ_PIPE,"Reading Delivery Pipe."},
    {ERR_INSUFFICIENT_MEMORY,"Insufficient memory."},
    {ERR_ODBC_SQLALLOCENV,"Cannot allocated ODBC env handle."},
```

```
    {ERR_SQL_ATTR_ODBC_VERSION,"Cannot set ODBC version."},
    {ERR_SQL_ATTR_CONNECTION_POOLING,"Cannot set Connection Pooling."},
    {0,""}
    };
    for(i=0; errorMsgs[i].szMsg[0]; i++)
    {
    if ( iError == errorMsgs[i].iError )
    {
    printf("\nError(%d): %s", iError, errorMsgs[i].szMsg);
    if ( fpLog )
    {
    EnterCriticalSection(&WriteLogCriticalSection);
    fprintf(fpLog, "*Error(%d): %s\r\n", iError, errorMsgs[i].szMsg);
    if ( bFlush )
    fflush(fpLog);
    LeaveCriticalSection(&WriteLogCriticalSection);
    }
    return;
    }
    }
    printf("Error(%d): Unknown Error.");
    EnterCriticalSection(&WriteLogCriticalSection);
    fprintf(fpLog, "*Error(%d): Unknown Error.\r\n", iError);
    if ( bFlush )
    fflush(fpLog);
    LeaveCriticalSection(&WriteLogCriticalSection);
    return;
}
/* FUNCTION: BOOL GetParameters(int argc, char *argv[])
 *
 * PURPOSE:This function parses the command line passed in to the
delivery executable, initializing
 *  and filling in global variable parameters.
 *
 * ARGUMENTS:intargcnumber of command line arguments passed to delivery
 *  char*argv[]array of command line argument pointers
 *
 * RETURNS:BOOLFALSEparameter read successfull
 *  TRUEuser has requested parameter information screen be displayed.
 *
 * COMMENTS:None
 *
 */
static BOOL GetParameters(int argc, char *argv[])
{
    int i;
    szServer[0]= 0;
    szPassword[0]= 0;
    bFlush= FALSE;
    strcpy(szDatabase, "tpcc");
    strcpy(szUser, "sa");
    for(i=0; i<argc; i++)
    {
    if ( argv[i][0] == '-' || argv[i][0] == '/' )
    {
    switch(argv[i][1])
    {
    case 'S':
    case 's':
    strcpy(szServer, argv[i]+2);
    break;
    case 'D':
    case 'd':
    strcpy(szDatabase, argv[i]+2);
    break;
    case 'U':
    case 'u':
    strcpy(szUser, argv[i]+2);
    break;
    case 'P':
    case 'p':
    strcpy(szPassword, argv[i]+2);
    break;
    case 'F':
    case 'f':
    bFlush = TRUE;//turn on delilog flush when written.
    break;
    case '?':
    return TRUE;
    }
    }
    }
    return FALSE;
}
/* FUNCTION: void PrintParameters(void)
 *
 * PURPOSE:This function displays the supported command line flags.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
```

```
 */
static void PrintParameters(void)
{
    PrintHeader();
    printf("DELISRV:\n\n");

printf("Parameter
Default\n");
    printf("-----------------------------------------------------------------
----------\n");
    printf("-S
Server                                                          \n");
    printf("-D
Database                                                tpcc    \n");
    printf("-U
Username                                                sa      \n");
    printf("-P
Password                                                        \n");
    printf("-F Flush output to delilog file when
written.                    OFF     \n");
    printf("-? This help screen\n\n");
    printf("Note:  Command line switches are NOT case sensitive.\n");
    return;
}
/* FUNCTION: void PrintHeader(void)
 *
 * PURPOSE:This function displays the delivery executable's banner
information.
 *
 * ARGUMENTS:None
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void PrintHeader(void)
{
    printf("************************************************\n");
    printf("*                                              *\n");
#ifdef USE_ODBC
    printf("*  Microsoft SQL Server 6.5 (ODBC)             *\n");
#else
    printf("*  Microsoft SQL Server 6.5 (DBLIB)            *\n");
#endif
    printf("*                                              *\n");
    printf("*  HTML TPC-C BENCHMARK KIT: Delivery Server    *\n");
    printf("*  Version %d.%2.2d.%3.3d                            *\n",
versionMS, versionMM, versionLS);
    printf("*                                              *\n");
    printf("************************************************\n\n");
    return;
}
/* FUNCTION: int ReadRegistrySettings(void)
 *
 * PURPOSE:This function reads the system registry filling in required
key parameters.
 *
 * ARGUMENTS:None
 *
 * RETURNS:intERR_REGISTRY_NOT_SETUPregistry not setup tpcc.exe needs to
be run
 *  to setup registry.
 *  ERR_SUCCESSReqistry read Successfull, no error
 *
 *
 * COMMENTS:None
 */
static int ReadRegistrySettings(void)
{
    HKEYhKey;
    DWORDsize;
    DWORDtype;
    charszTmp[256];
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\TPCC",
0, KEY_READ, &hKey) != ERROR_SUCCESS )
    return ERR_REGISTRY_NOT_SETUP;
    size = sizeof(szTmp);
    iNumThreads = 4;
    if ( RegQueryValueEx(hKey, "NumberOfDeliveryThreads", 0, &type,
szTmp, &size) == ERROR_SUCCESS )
    iNumThreads = atoi(szTmp);
    if ( !iNumThreads )
    iNumThreads = 4;
    iDelayMs = 1000;
    if ( RegQueryValueEx(hKey, "BackoffDelay", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iDelayMs = atoi(szTmp);
    if ( !iDelayMs )
    iDelayMs = 1000;
    iDeadlockRetry = 3;
    if ( RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iDeadlockRetry = atoi(szTmp);
    if ( !iDeadlockRetry )
    iDeadlockRetry = 3;
    RegCloseKey(hKey);
```

```
    return ERR_SUCCESS;
}
/* FUNCTION: void CheckKey(void *ptr)
 *
 * PURPOSE:This function checks for a key press on the delivery
executable's console. If the
 *  key press is a Ctrl C then the execution termination flag variable
bDone is set to
 *  TRUE which will start the termination of the delivery executable.
 *
 * ARGUMENTS:void*ptrdummy argument passed in though thread manager,
unused NULL.
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void CheckKey(void *ptr)
{
    while( _getch() != CTRL_C)
    ;
    bDone = TRUE;
    return;
}
/* FUNCTION: void DeliveryHandler( void *ptr )
 *
 * PURPOSE:This function is executed in it's own thread what it does is
to check for delivery
 *  postings in the delivery named pipe. If any are present then it
pulls them off and
 *  places them in the next available delivery queue array element.
 *
 * ARGUMENTS:void*ptrdummy argument passed in though thread manager,
unused NULL.
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void DeliveryHandler( void *ptr )
{
    inti;
    intsize;
    intiError;
    while( !bDone )
    {
    for(i=0; i<iQSlotts; i++)
    {
    if ( !pDeliveryCache[i].bInUse )
    break;
    }
    if ( i < iQSlotts )
    {
    EnterCriticalSection(&DeliveryCriticalSection);
    pDeliveryCache[i].bInUse = TRUE;
    LeaveCriticalSection(&DeliveryCriticalSection);
    }
    else
    {
    EnterCriticalSection(&DeliveryCriticalSection);
    if ( !(pDeliveryCache = (LPDELIVERY_PACKET)realloc(pDeliveryCache,
sizeof(DELIVERY_PACKET) * (iQSlotts+512))) )
    {
    ErrorMessage(ERR_INSUFFICIENT_MEMORY);
    LeaveCriticalSection(&DeliveryCriticalSection);
    return;
    }
    for(i=iQSlotts; i<iQSlotts+512; i++)
    pDeliveryCache[i].bInUse = FALSE;
    i = iQSlotts;
    pDeliveryCache[i].bInUse = TRUE;
    LeaveCriticalSection(&DeliveryCriticalSection);
    }
    pDeliveryCache[i].ov.Offset= i;
    pDeliveryCache[i].ov.Internal= 0;
    pDeliveryCache[i].ov.InternalHigh= 0;
    pDeliveryCache[i].ov.OffsetHigh= 1;
    pDeliveryCache[i].ov.hEvent= NULL;
    while( !bDone )
    {
    if ( ReadFile(hPipe, &pDeliveryCache[i].trans,
sizeof(DELIVERY_TRANSACTION), &size, &pDeliveryCache[i].ov) )
    break;
    if ( bDone )
    break;
    iError = GetLastError();
    if ( iError == ERROR_IO_PENDING )
    {
    while( pDeliveryCache[i].ov.OffsetHigh )
    Sleep(10);
    break;
    }
    else
```

```
    {
    ErrorMessage(ERR_READ_PIPE);
    return;
    }
    }
    Sleep(1);
    }
    return;
}
/* FUNCTION: void DeliveryThread( void *ptr )
 *
 * PURPOSE:This function is executed inside the delivery threads. The
queue array
 *  is continuously check and if any array elements are in use then the
 *  array entry is read, cleared and this function processes it.
 *
 * ARGUMENTS:void*ptrdummy argument passed in though thread manager,
unused NULL.
 *
 * RETURNS:None
 *
 * COMMENTS:The registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
 *  value NumberOfDeliveryThreads controls how many of these
 *  functions are running. The HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
 *  value BackoffDelay controls the amount of time this function waits
 *  between checks of the delivery queue.
 *
 */
static void DeliveryThread( void *ptr )
{
    intsize;
    intkey;
    LPOVERLAPPEDpov;
    DELIVERYdelivery;
    intiError;
    if ( SQLOpenConnection(&delivery.dbproc, szServer, szDatabase,
szUser, szPassword, &delivery.spid) )
    return;//error posting tbd
    //while delisrv running i.e. user has not requested termination
    while( !bDone )
    {
    if ( GetQueuedCompletionStatus(hComPort, &size, &key, &pov, (DWORD)-
1) )
    {
    pov->OffsetHigh = 0;//clear to notify delivery handler ok to read
another entry.
    //some delivery to do so process it
    memcpy(&delivery.queue, &pDeliveryCache[pov->Offset].trans.queue,
sizeof(SYSTEMTIME));
    delivery.w_id= pDeliveryCache[pov->Offset].trans.w_id;
    delivery.o_carrier_id= pDeliveryCache[pov-
>Offset].trans.o_carrier_id;
    if ( (iError=SQLDelivery(&delivery)) )
    {
    ErrorMessage(iError);
    printf("Running : ");
    continue;
    }
    //update log
    WriteLog(&delivery);
    EnterCriticalSection(&DeliveryCriticalSection);
    pDeliveryCache[pov->Offset].bInUse = FALSE;
    LeaveCriticalSection(&DeliveryCriticalSection);
    }
    }
    return;
}
/* FUNCTION: static int err_handler(DBPROCESS *dbproc, int severity, int
dberr, int oserr, char *dberrstr, char *oserrstr)
 *
 * PURPOSE:This function handles DB-Library errors
 *
 * ARGUMENTS:DBPROCESS*dbprocDBPROCESS id pointer
 *  intseverityseverity of error
 *  intdberrerror id
 *  intoserroperating system specific error code
 *  char*dberrstrprintable error description of dberr
 *  char*oserrstrprintable error description of oserr
 *
 * RETURNS:intINT_CONTINUEcontinue if error is SQLETIME else INT_CANCEL
action
 *
 * COMMENTS:None
 *
 */
#ifndef USE_ODBC
static int err_handler(DBPROCESS *dbproc, int severity, int dberr, int
oserr, char *dberrstr, char *oserrstr)
{
    if (oserr != DBNOERR)
    printf("(%d) %s", oserr, oserrstr);
    if ((dbproc == NULL) || (DBDEAD(dbproc)))
    ExitThread((unsigned long)-1);
    return INT_CONTINUE;
}
```

```
#endif
/* FUNCTION: static int msg_handler(DBPROCESS *dbproc, DBINT msgno, int
msgstate, int severity, char *msgtext)
 *
 * PURPOSE:This function handles DB-Library SQL Server error messages
 *
 * ARGUMENTS:DBPROCESS*dbprocDBPROCESS id pointer
 *  DBINTmsgnomessage number
 *  intmsgstatemessage state
 *  intseveritymessage severity
 *  char*msgtextprintable message description
 *
 * RETURNS:intINT_CONTINUEcontinue if error is SQLETIME else INT_CANCEL
action
 *  INT_CANCELcancel operation
 *
 * COMMENTS:This function also sets the dead lock dbproc variable if
necessary.
 *
 */
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**dbprocpointer to returned DBPROCESS
 *  char*serverSQL server name
 *  char*databaseSQL server database
 *  char*useruser name
 *  char*passworduser password
 *  int*spidpointer to returned spid
 *
 * RETURNS:BOOLFALSEif successfull
 *  TRUEif an error occurs
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    static BOOL SQLOpenConnection(DBPROCESS **dbproc, char *server, char
*database, char *user, char *password, int *spid)
    {
    RETCODErc;
    charbuffer[30];
    *dbproc = (DBPROCESS *)malloc(sizeof(DBPROCESS));
    if ( !*dbproc )
    return TRUE;
    //set pECB data into dbproc
    dbsetuserdata(*dbproc, malloc(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)DEFCLPACKSIZE);
    if ((*dbproc = dbopen(login, server )) == NULL)
    return TRUE;

    // Use the the right database
    dbuse(*dbproc, database);
    dbsetuserdata(*dbproc, malloc(sizeof(BOOL)));
    *((BOOL *)dbgetuserdata(*dbproc)) = FALSE;
    dbcmd(*dbproc, "select @@spid");
    dbsqlexec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
    dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *) spid);
    while (dbnextrow(*dbproc) != NO_MORE_ROWS);
    }
    dbcmd(*dbproc, "set nocount on");
    dbsqlexec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    while (dbnextrow(*dbproc) != NO_MORE_ROWS);
    return FALSE;
    }
#endif
//queue time, end time, elapsed time, w_id, o_carrier_id, o_id1, ...
o_id10
/* FUNCTION: void WriteLog(LPDELIVERY pDelivery)
 *
 * PURPOSE:This function writes the delivery results to the delivery log
file.
 *
 * ARGUMENTS:LPDELIVERYpDeliveryPointer to delivery information.
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void WriteLog(LPDELIVERY pDelivery)
{
    int elapsed;
    CalculateElapsedTime(&elapsed, &pDelivery->queue, &pDelivery-
>trans_end);
    EnterCriticalSection(&WriteLogCriticalSection);
    fprintf(fpLog,
"%2.2d/%2.2d/%2.2d,%2.2d:%2.2d:%2.2d:%3.3d,%2.2d:%2.2d:%2.2d:%3.3d,%d,%d,
%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
    pDelivery->trans_end.wYear - 1900, pDelivery->trans_end.wMonth,
pDelivery->trans_end.wDay,
    pDelivery->queue.wHour, pDelivery->queue.wMinute, pDelivery-
>queue.wSecond, pDelivery->queue.wMilliseconds,
    pDelivery->trans_end.wHour, pDelivery->trans_end.wMinute, pDelivery-
>trans_end.wSecond, pDelivery->trans_end.wMilliseconds,
    elapsed,
    pDelivery->w_id, pDelivery->o_carrier_id,
    pDelivery->o_id[0], pDelivery->o_id[1], pDelivery->o_id[2],
pDelivery->o_id[3],
    pDelivery->o_id[4], pDelivery->o_id[5], pDelivery->o_id[6],
pDelivery->o_id[7],
    pDelivery->o_id[8], pDelivery->o_id[9] );
    if ( bFlush )
    fflush(fpLog);
    LeaveCriticalSection(&WriteLogCriticalSection);
    return;
}
/* FUNCTION: void CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME
lpBegin, LPSYSTEMTIME lpEnd)
 *
 * PURPOSE:This function calculates the elapsed time a delivery
transaction took.
 *
 * ARGUMENTS:int*pElapsedpointer to int variable to receive calculated
elapsed
 *  time in milliseconds.
 *  LPSYSTEMTIMElpBeginPointer to system time structure containing
 *  transaction beginning time.
```

```
 *  LPSYSTEMTIMElpEndPointer to system time structure containing
 *  transaction ending time.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME lpBegin,
LPSYSTEMTIME lpEnd)
{
    intbeginSeconds;
    intendSeconds;
    beginSeconds = (lpBegin->wHour * 3600000) + (lpBegin->wMinute *
60000) + (lpBegin->wSecond * 1000) + lpBegin->wMilliseconds;
    endSeconds = (lpEnd->wHour * 3600000) + (lpEnd->wMinute * 60000) +
(lpEnd->wSecond * 1000) + lpEnd->wMilliseconds;
    *pElapsed = endSeconds - beginSeconds;
    //check for day boundry, this will function for 24 hour period
however it will not work over 48 hours.
    if ( *pElapsed < 0 )
    *pElapsed = *pElapsed + (24 * 60 * 60 * 1000);
    return;
}
/* FUNCTION: int SQLDelivery(DELIVERY *pDelivery)
 *
 * PURPOSE:This function processes the delivery transaction.
 *
 * ARGUMENTS:DELIVERY*pDeliveryPointer to delivery transaction structure
 *
 * RETURNS:intERR_DBGETDATA_FAILEDDelivery get data operation failed.
 *  ERR_SUCCESSDelivery successfull, no error
 *
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    static int SQLDelivery(DELIVERY *pDelivery)
    {
    inti;
    intdeadlock_count;
    SDWORD  iLength[10];
    BOOLbDeadlock;
    deadlock_count = 0;
    // Start new delivery
    while ( TRUE )
    {
    BindParameter(pDelivery->dbproc, 1, SQL_C_SSHORT, SQL_SMALLINT, 0,
0, &pDelivery->w_id, 0);
    BindParameter(pDelivery->dbproc, 2, SQL_C_SSHORT, SQL_SMALLINT, 0,
0, &pDelivery->o_carrier_id, 0);
    if ( ExecuteStatement(pDelivery->dbproc, "{call tpcc_delivery (?,
?)}") )
    return 1;
    bDeadlock = *((BOOL *)dbgetuserdata(pDelivery->dbproc));
    if ( !bDeadlock )
    {
    for (i=0;i<10;i++)
    {
    if ( BindColumn(pDelivery->dbproc, (UWORD)(i+1), SQL_C_SLONG,
&pDelivery->o_id[i], 0, &iLength[i]) )
    return 1;
    }
    if ( GetResults(pDelivery->dbproc) )
    return 1;
    for(i=0; i<10; i++)
    {
    if ( iLength[i] <= 0 )
    pDelivery->o_id[i] = 0;
    }
    }
    SQLFreeStmt(pDelivery->dbproc->hstmt, SQL_CLOSE);
    if ( !SQLDetectDeadlock(pDelivery->dbproc) )
    break;
    deadlock_count++;
    Sleep(10 * deadlock_count);
    }
    GetLocalTime(&pDelivery->trans_end);
    return ERR_SUCCESS;
    }
#else
    static int SQLDelivery(DELIVERY *pDelivery)
    {
    RETCODErc;
    inti;
    intdeadlock_count;
    BYTE*pData;
    deadlock_count = 0;
    // Start new delivery
    while ( TRUE )
    {
    if (dbrpcinit(pDelivery->dbproc, "tpcc_delivery", 0) == SUCCEED)
    {
    dbrpcparam(pDelivery->dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE
*)&pDelivery->w_id);
```

```
    dbrpcparam(pDelivery->dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *)
&pDelivery->o_carrier_id);

    if (dbrpcexec(pDelivery->dbproc) == SUCCEED)
    {
    while (((rc = dbresults(pDelivery->dbproc)) != NO_MORE_RESULTS) &&
(rc != FAIL))
    {
    while (((rc = dbnextrow(pDelivery->dbproc)) != NO_MORE_ROWS) && (rc
!= FAIL))
    {
    for (i=0;i<10;i++)
    {
    if(pData=dbdata(pDelivery->dbproc, i+1))
    pDelivery->o_id[i] = *((DBINT *)pData);
    else
    pDelivery->o_id[i] = 0;
    }
    }
    }
    }
    }
    if ( !SQLDetectDeadlock(pDelivery->dbproc) )
    break;
    deadlock_count++;
    Sleep(10 * deadlock_count);
    }
    GetLocalTime(&pDelivery->trans_end);
    return ERR_SUCCESS;
    }
#endif
/* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
 *
 * PURPOSE:This function is used to check for deadlock conditions.
 *
 * ARGUMENTS:DBPROCESS*dbprocDBPROCESS to check
 *
 * RETURNS:BOOLFALSENo lock condition present
 *  TRUELock condition detected
 *
 * COMMENTS:None
 *
 */
static BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
{
    if (*((BOOL *) dbgetuserdata(dbproc)) == TRUE)
    {
    *((BOOL *) dbgetuserdata(dbproc)) = FALSE;
    return TRUE;
    }
    return FALSE;
}
/* FUNCTION: int OpenLogFile(void)
 *
 * PURPOSE:This function opens the delivery log file for use.
 *
 * ARGUMENTS:None
 *
 * RETURNS:intERR_REGISTRY_NOT_SETUPRegistry not setup.
 *  ERR_CANNOT_CREATE_RESULTS_FILECannot create results log file.
 *  ERR_SUCCESSLog file successfully opened
 *
 *
 * COMMENTS:None
 *
 */
static int OpenLogFile(void)
{
    HKEYhKey;
    BOOLbRc;
    BYTEszTmp[256];
    charszKey[256];
    charszLogPath[256];
    DWORDsize;
    DWORDsv;
    intlen;
    char*ptr;
    szLogPath[0] =  0;
    bRc = TRUE;
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parameters\\Virtual Roots",
0, KEY_ALL_ACCESS, &hKey) == ERROR_SUCCESS )
    {
    sv = sizeof(szKey);
    size = sizeof(szTmp);
    if ( RegEnumValue(hKey, 0, szKey, &sv, NULL, NULL, szTmp, &size) ==
ERROR_SUCCESS )
    {
    strcpy(szLogPath, szTmp);
    bRc = FALSE;
    }
    RegCloseKey(hKey);
    }
    if ( bRc )
```

```
    return ERR_REGISTRY_NOT_SETUP;
    if ( (ptr = strchr(szLogPath, ',')) )
    *ptr = 0;
    len = strlen(szLogPath);
    if ( szLogPath[len-1] != '\\' )
    {
    szLogPath[len] = '\\';
    szLogPath[len+1] = 0;
    }
    strcat(szLogPath, "delilog.");
    fpLog = fopen(szLogPath, "ab");
    if ( !fpLog )
    return ERR_CANNOT_CREATE_RESULTS_FILE;
    return ERR_SUCCESS;
}
#ifdef USE_ODBC
    /* FUNCTION: void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
     *
     * PURPOSE:This function sets a user pointer in a dbproc structure
     *This functionality is not provided in odbc so this function
     *provides it.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *void *uPtrreturned data user pointer
     *
     * RETURNS:none
     *
     * COMMENTS:The caller is responsible for the contents of the uPtr.
     *
     */
    void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
    {
    dbproc->uPtr = uPtr;
    }
    /* FUNCTION: void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
     *
     * PURPOSE:This function returns the user pointer stored in a dbproc
structure
     *This functionality is not provided in odbc so this function
     *provides it.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:none
     *
     * COMMENTS:The returned pointer is placed in the dbproc structure
by the dbsetuserdata() API.
     *
     */
    void *dbgetuserdata(PDBPROCESS dbproc)
    {
    return dbproc->uPtr;
    }
    /* FUNCTION: void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD
fCType, SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue,
SDWORD cbValueMax)
     *
     * PURPOSE:This function wraps the functionality provided by the
SQLBindParameter
     *allowing error process so that each bind call does not need to
provide
     *error and message checking.
     *
     * ARGUMENTS:PDBPROCESSdbprocpointer to odbc dbprocess structure
     *UWORDiparParameter number, ordered sequentially left to right,
starting at 1.
     *SWORDfParamTypeThe type of the parameter.
     *SWORDfCTypeThe C data type of the parameter.
     *SWORDfSqlTypeThe SQL data type of the parameter.
     *UDWORDcbColDefThe precision of the column or expression
     *of the corresponding parameter marker.
     *SWORDibScaleThe scale of the column or expression of the
corresponding
     *parameter marker.
     *PTRrgbValueA pointer to a buffer for the parameter's data.
     *SDWORDcbValueMaxMaximum length of the rgbValue buffer.
     *void *uPtrreturned data user pointer
     *
     * RETURNS:none
     *
     * COMMENTS:The returned pointer is placed in the dbproc structure
by the dbset
     *
     */
    void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD fCType,
SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue, SDWORD
cbValueMax)
    {
    RETCODE rc;
    rc = SQLBindParameter(dbproc->hstmt, ipar, SQL_PARAM_INPUT, fCType,
fSqlType, cbColDef, ibScale, rgbValue, cbValueMax, NULL);
    if (rc == SQL_ERROR)
    ODBCError(dbproc);
    return;
    }
    /* FUNCTION: void ODBCError(PDBPROCESS dbproc)
```

```
     *
     * PURPOSE:This function wraps the odbc error call so that the dblib
msg_handler is called.
     *This allows the deadlock flag in the dbproc user data structure
pEcbInfo in
     *dbproc to be set if necessary.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    void ODBCError(PDBPROCESS dbproc)
    {
    SDWORDlNativeError;
    charszState[6];
    charszMsg[SQL_MAX_MESSAGE_LENGTH];
    while( SQLError(henv, dbproc->hdbc, dbproc->hstmt, szState,
&lNativeError, szMsg, sizeof(szMsg), NULL) == SQL_SUCCESS )
    {
    msg_handler(dbproc, lNativeError, 0, 0, szMsg);
    if ( !lNativeError )
    {
    printf("\nODBC Error State = %s, %s\n", szState, szMsg);
    printf("Running : ");
    }
    }
    return;
    }
    /* FUNCTION: BOOL ExecuteStatement(PDBPROCESS dbproc, szStatement)
     *
     * PURPOSE:This function wraps the odbc SQLExecDirect API so that
error handling and
     *and deadlock are taken care of in a common location.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *char*szStatementsql stored procedure statement to be executed.
     *
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement)
    {
    RETCODErc;
    rc = SQLExecDirect(dbproc->hstmt, szStatement, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
    {
    ODBCError(dbproc);
    if ( *((BOOL *)dbgetuserdata(dbproc)) )
    return FALSE;
    return TRUE;
    }
    return FALSE;
    }
    /* FUNCTION: BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol,
SQLSMALLINT fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax, SDWORD
FAR *piLength)
     *
     * PURPOSE:This function wraps the odbc SQLBindCol API so that error
handling and
     *and deadlock are taken care of in a common location.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *UWORDicolColumn number of result data, ordered sequentially left
to right, starting at 1.
     *SWORDfCTypeThe C data type of the result data. SQL_C_BINARY,
SQL_C_BIT, SQL_C_BOOKMARK,
     *SQL_C_CHAR, SQL_C_DATE, SQL_C_DEFAULT, SQL_C_DOUBLE, SQL_C_FLOAT,
SQL_C_SLONG,
     *SQL_C_SSHORT, SQL_C_STINYINT, SQL_C_TIME, SQL_C_TIMESTAMP,
SQL_C_ULONG,
     *SQL_C_USHORT, SQL_C_UTINYINT, SQL_C_DEFAULT
     *PTRrgbValuePointer to storage for the data. If rgbValue is a null
pointer, the
     *driver unbinds the column.
     *SDWORDcbValueMaxMaximum length of the rgbValue buffer. For
character data, rgbValue
     *must also include space for the null-termination byte.
     *SDWORD *piLengthPointer to variable to receive length of returned
data.
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol, SQLSMALLINT
fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax, SDWORD *piLength)
    {
    RETCODErc;
    rc = SQLBindCol(dbproc->hstmt, icol, fCType, rgbValue, cbValueMax,
piLength);
    if ( rc == SQL_ERROR )
```

```
    {
    ODBCError(dbproc);
    return TRUE;
    }
    return FALSE;
    }
    /* FUNCTION: BOOL GetResults(PDBPROCESS dbproc)
     *
     * PURPOSE:This function wraps the odbc SQLFetch API so that error
handling and
     *and deadlock are taken care of in a common location.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    BOOL GetResults(PDBPROCESS dbproc)
    {
    if ( SQLFetch(dbproc->hstmt) == SQL_ERROR )
    {
    ODBCError(dbproc);
    if ( *((BOOL *)dbgetuserdata(dbproc)) )
    return FALSE;
    return TRUE;
    }
    return FALSE;
    }
    /* FUNCTION: BOOL MoreResults(DBPROCESS  dbproc)
     *
     * PURPOSE:This function wraps the odbc SQLMoreResults API so that
error handling and
     *and deadlock are taken care of in a common location.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    BOOL MoreResults(PDBPROCESS  dbproc)
    {
    if ( SQLMoreResults(dbproc->hstmt) == SQL_ERROR )
    {
        ODBCError(dbproc);
        if ( *((BOOL *)dbgetuserdata(dbproc)) )
        return FALSE;
        return TRUE;
        }
        return FALSE;
        }
    #endif
```

Delisrv.h

```
    /*  FILE:DELISRV.H, MSTPCC.300
     *  ------------------------------------------------
     *  Microsoft TPC-C Kit Ver. 3.00.000
     *  Audited 08/23/96, By Francois Raab
     *
     *  Copyright Microsoft, 1996
     *
     *  PURPOSE:Header file for delivery service executable
     *  Author:Philip Durr
     *  philipdu@Microsoft.com
     */
    #define AVAILABLE0//queue array element available
    #define WRITE_LOCKED1//queue array element is being written to
    #define READ_LOCKED2//queue array element is begin read
    #defineINUSE4//queue array element has information stored in it
    #defineCTRL_C3//<Ctrl> C, exit key code
    #define DEFCLPACKSIZE4096//default DB Library SQL Connection pack size
    #defineERR_SUCCESS0//Success, no error.
    #define ERR_CANNOT_CREATE_THREAD1000//Cannot create thread.
    #define ERR_DBGETDATA_FAILED1001//Get data failed.
    #defineERR_REGISTRY_NOT_SETUP1002//Registry not setup for tpcc.
    #defineERR_CANNOT_ACCESS_DELIVERY_FN1003//Cannot access ReadDelivery
    cache.
    #defineERR_CANNOT_ACCESS_REGISTRY1004//Cannot access registry key TPCC.
    #defineERR_CANNOT_CREATE_RESULTS_FILE1005//Cannot create results file.
    #defineERR_CANNOT_OPEN_PIPE1006//Cannot open delivery pipe.
    #defineERR_READ_PIPE1007//Error reading pipe
    #defineERR_INSUFFICIENT_MEMORY1008//insufficient memory
    #defineERR_ODBC_SQLALLOCENV1009//Cannot allocated ODBC env handle
    #defineERR_SQL_ATTR_ODBC_VERSION1010//Cannot set ODBC version
    #defineERR_SQL_ATTR_CONNECTION_POOLING1011//Cannot set Connection Pooling
    typedef struct _DELIVERY_TRANSACTION
    {
        SYSTEMTIMEqueue;//time delivery transaction queued
```

```
    shortw_id;//delivery warehouse
    shorto_carrier_id;//carrier id
} DELIVERY_TRANSACTION;
typedef DELIVERY_TRANSACTION *LPDELIVERY_TRANSACTION;//pointer to
delivery transaction queue
typedef struct _DELIVERY_PACKET
{
    BOOLbInUse;//entry current in use
    OVERLAPPEDov;//pipe io overlapped structure
    DELIVERY_TRANSACTIONtrans;//delivery transaction information
} DELIVERY_PACKET, *LPDELIVERY_PACKET;
typedef struct _SERRORMSG
{
    intiError;//error mesage id
    charszMsg[80];//error message
} SERRORMSG;
#ifdef USE_ODBC
    typedef struct _DBPROCESS
    {
    HDBChdbc;
    HSTMThstmt;
    intspid;
    void*uPtr;
    } DBPROCESS, *PDBPROCESS;
    //dblib error message return values
    #define INT_EXIT        0
    #define INT_CONTINUE    1
    #define INT_CANCEL      2
#endif
//delivery transaction structure
typedef struct DELIVERY
{
    shortw_id;//warehouse id
    shorto_carrier_id;//carrier id
    intspid;//db library spid
    longo_id[10];//returned delivery transaction ids
    DBPROCESS*dbproc;//db library DBPROCESS pointer
    SYSTEMTIMEqueue;//delivery transaction queue time
    SYSTEMTIMEtrans_end;//delivery transaction finished time
} DELIVERY;
typedef DELIVERY *LPDELIVERY;//pointer to delivery structure
//function prototypes
voidmain(int argc, char *argv[]);
staticvoidcls(void);
staticintRunDelivery(void);
staticvoidQuitStatus(void);
staticvoidAnimateWait1(void);
staticvoidAnimateWait(void);
staticintInit(void);
staticvoidRestore(void);
staticvoidErrorMessage(int iError);
staticBOOLGetParameters(int argc, char *argv[]);
staticvoidPrintParameters(void);
staticvoidPrintHeader(void);
staticintReadRegistrySettings(void);
staticvoidCheckKey(void *ptr);
staticvoidDeliveryHandler( void *ptr );
staticvoidDeliveryThread( void *ptr );
#ifndef USE_ODBC
    staticinterr_handler(DBPROCESS *dbproc, int severity, int dberr, int
oserr, char *dberrstr, char *oserrstr);
#endif
#ifdef USE_ODBC
    #define DBINTint
#endif
staticintmsg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext);
staticBOOLSQLOpenConnection(DBPROCESS **dbproc, char *server, char
*database, char *user, char *password, int *spid);
staticvoidWriteLog(LPDELIVERY pDelivery);
staticvoidCalculateElapsedTime(int *pElapsed, LPSYSTEMTIME lpBegin,
LPSYSTEMTIME lpEnd);
staticintSQLDelivery(DELIVERY *pDelivery);
staticBOOLSQLDetectDeadlock(DBPROCESS *dbproc);
staticBOOLReadDeliveryInfo(short *w_id, short *o_carrier_id);
staticBOOLPostDeliveryInfo(short w_id, short o_carrier_id);
staticintOpenLogFile(void);
#ifdef USE_ODBC
    void dbsetuserdata(PDBPROCESS dbproc, void *uPtr);
    void *dbgetuserdata(PDBPROCESS dbproc);
    void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD fCType,
SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue, SDWORD
cbValueMax);
    void ODBCError(PDBPROCESS dbproc);
    BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement);
    BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol, SQLSMALLINT
fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax, SDWORD *piLength);
    BOOL GetResults(PDBPROCESS dbproc);
    BOOL MoreResults(PDBPROCESS  dbproc);
    BOOL ReopenConnection(PDBPROCESS dbproc);
#endif
```

error.c

```
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include "trans.h"
#include "tpcc.h"
#include "util.h"
#include "error.h"
/* FUNCTION: void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int
iError, int iErrorType, char *szMsg)
 *
 * PURPOSE:This function displays an error message in the client browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiErrorid of error message
 *  intiErrorTypeerror type, ERR_TYPE_SQL, ERR_TYPE_DBLIB, or
ERR_TYPE_WEBDLL
 *  intiTermIdterminal id from browser
 *  intiSyncidsync id from browser
 *  char *szMsgoptional error message string used with ERR_TYPE_SQL and
 *  ERR_TYPE_DBLIB
 *
 * RETURNS:None
 *
 * COMMENTS:If the error type is ERR_TYPE_WEBDLL the szmsg parameter may
be NULL because it
 *  is ignored. If the error type is ERR_TYPE_SQL or ERR_TYPE_DBLIB then
the szMsg
 *  parameter contains the text of the error message, so the szMsg
parameter cannot
 *  be NULL.
 *
 */
void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType, char *szMsg, int iTermId, int iSyncId)
{
    int i;
    static SERRORMSG errorMsgs[] =
    {
    {ERR_SUCCESS,"Success, no error."},
    {ERR_COMMAND_UNDEFINED,"Command undefined."},
    {ERR_NOT_IMPLEMENTED_YET,"Not Implemented Yet."},
    {ERR_CANNOT_INIT_TERMINAL,"Cannot initialize client connection."},
    {ERR_OUT_OF_MEMORY,"insufficient memory."},
    {ERR_NEW_ORDER_NOT_PROCESSED,"Cannot process new Order form."},
    {ERR_PAYMENT_NOT_PROCESSED,"Cannot process payment form."},
    {ERR_NO_SERVER_SPECIFIED,"No Server name specified."},
    {ERR_ORDER_STATUS_NOT_PROCESSED,"Cannot process order status form."},
    {ERR_W_ID_INVALID,"Invalid Warehouse ID."},
    {ERR_CAN_NOT_SET_MAX_CONNECTIONS,"Insufficient memory to allocate #
connections."},
    {ERR_NOSUCH_CUSTOMER,"No such customer."},
    {ERR_D_ID_INVALID,"Invalid District ID Must be 1 to 10."},
    {ERR_MAX_CONNECT_PARAM,"Max client connections exceeded, run install
to increase."},
    {ERR_INVALID_SYNC_CONNECTION,"Invalid Terminal Sync ID."},
    {ERR_INVALID_TERMID,"Invalid Terminal ID."},
    {ERR_PAYMENT_INVALID_CUSTOMER,"Payment Form, No such Customer."},
    {ERR_SQL_OPEN_CONNECTION,"SQLOpenConnection API Failed."},
    {ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,"Stock Level missing Threshold
key \"TT*\"."},
    {ERR_STOCKLEVEL_THRESHOLD_INVALID,"Stock Level Threshold invalid
data type range = 1 - 99."},
    {ERR_STOCKLEVEL_THRESHOLD_RANGE,"Stock Level Threshold out of range,
range must be 1 - 99."},
    {ERR_STOCKLEVEL_NOT_PROCESSED,"Stock Level not processed."},
    {ERR_NEWORDER_FORM_MISSING_DID,"New Order missing District key
\"DID*\"."},
    {ERR_NEWORDER_DISTRICT_INVALID,"New Order District ID Invalid range
1 - 10."},
    {ERR_NEWORDER_DISTRICT_RANGE,"New Order District ID out of Range.
Range = 1 - 10."},
    {ERR_NEWORDER_CUSTOMER_KEY,"New Order missing Customer key \"CID*\"."
},
    {ERR_NEWORDER_CUSTOMER_INVALID,"New Order customer id invalid data
type, range = 1 to 3000."},
    {ERR_NEWORDER_CUSTOMER_RANGE,"New Order customer id out of range,
range = 1 to 3000."},
    {ERR_NEWORDER_MISSING_IID_KEY,"New Order missing Item Id key
\"IID*\"."},
    {ERR_NEWORDER_ITEM_BLANK_LINES,"New Order blank order lines all
orders must be continuous."},
    {ERR_NEWORDER_ITEMID_INVALID,"New Order Item Id is wrong data type,
must be numeric."},
    {ERR_NEWORDER_MISSING_SUPPW_KEY,"New Order missing Supp_W key
\"SP##*\"."},
    {ERR_NEWORDER_SUPPW_INVALID,"New Order Supp_W invalid data type must
be numeric."},
    {ERR_NEWORDER_MISSING_QTY_KEY,"New Order Missing Qty key \"Qty##*\"."
},
    {ERR_NEWORDER_QTY_INVALID,"New Order Qty invalid must be numeric
range 1 - 99."},
    {ERR_NEWORDER_SUPPW_RANGE,"New Order Supp_W value out of range range
= 1 - Max Warehouses."},
    {ERR_NEWORDER_ITEMID_RANGE,"New Order Item Id is out of range. Range
= 1 to 999999."},
    {ERR_NEWORDER_QTY_RANGE,"New Order Qty is out of range. Range = 1 to
99."},
```

```
    {ERR_PAYMENT_DISTRICT_INVALID,"Payment District ID is invalid must
be 1 - 10."},
    {ERR_NEWORDER_SUPPW_WITHOUT_ITEMID,"New Order Supp_W field entered
without a corrisponding Item_Id."},
    {ERR_NEWORDER_QTY_WITHOUT_ITEMID,"New Order Qty entered without a
corrisponding Item_Id."},
    {ERR_NEWORDER_NOITEMS_ENTERED,"New Order Blank Items between items,
items must be continuous."},
    {ERR_PAYMENT_MISSING_DID_KEY,"Payment missing District Key \"DID*\"."
},
    {ERR_PAYMENT_DISTRICT_RANGE,"Payment District Out of range, range =
1 - 10."},
    {ERR_PAYMENT_MISSING_CID_KEY,"Payment missing Customer Key \"CID*\"."
},
    {ERR_PAYMENT_CUSTOMER_INVALID,"Payment Customer data type invalid,
must be numeric."},
    {ERR_PAYMENT_MISSING_CLT,"Payment missing Customer Last Name Key
\"CLT*\"."},
    {ERR_PAYMENT_LAST_NAME_TO_LONG,"Payment Customer last name longer
than 16 characters."},
    {ERR_PAYMENT_CUSTOMER_RANGE,"Payment Customer ID out of range, must
be 1 to 3000."},
    {ERR_PAYMENT_CID_AND_CLT,"Payment Customer ID and Last Name entered
must be one or other."},
    {ERR_PAYMENT_MISSING_CDI_KEY,"Payment missing Customer district key
\"CDI*\"."},
    {ERR_PAYMENT_CDI_INVALID,"Payment Customer district invalid must be
numeric."},
    {ERR_PAYMENT_CDI_RANGE,"Payment Customer district out of range must
be 1 - 10."},
    {ERR_PAYMENT_MISSING_CWI_KEY,"Payment missing Customer Warehouse key
\"CWI*\"."},
    {ERR_PAYMENT_CWI_INVALID,"Payment Customer Warehouse invalid must be
numeric."},
    {ERR_PAYMENT_CWI_RANGE,"Payment Customer Warehouse out of range, 1
to Max Warehouses."},
    {ERR_PAYMENT_MISSING_HAM_KEY,"Payment missing Amount key \"HAM*\"."},
    {ERR_PAYMENT_HAM_INVALID,"Payment Amount invalid data type must be
numeric."},
    {ERR_PAYMENT_HAM_RANGE,"Payment Amount out of range, 0 - 9999.99."},
    {ERR_ORDERSTATUS_MISSING_DID_KEY,"Order Status missing District key
\"DID*\"."},
    {ERR_ORDERSTATUS_DID_INVALID,"Order Status District invalid, value
must be numeric 1 - 10."},
    {ERR_ORDERSTATUS_DID_RANGE,"Order Status District out of range must
be 1 - 10."},
    {ERR_ORDERSTATUS_MISSING_CID_KEY,"Order Status missing Customer key
\"CID*\"."},
    {ERR_ORDERSTATUS_MISSING_CLT_KEY,"Order Status missing Customer Last
Name key \"CLT*\"."},
    {ERR_ORDERSTATUS_CLT_RANGE,"Order Status Customer last name longer
than 16 characters."},
    {ERR_ORDERSTATUS_CID_INVALID,"Order Status Customer ID invalid,
range must be numeric 1 - 3000."},
    {ERR_ORDERSTATUS_CID_RANGE,"Order Status Customer ID out of range
must be 1 - 3000."},
    {ERR_ORDERSTATUS_CID_AND_CLT,"Order Status Customer ID and LastName
entered must be only one."},
    {ERR_DELIVERY_MISSING_OCD_KEY,"Delivery missing Carrier ID key
\"OCD*\"."},
    {ERR_DELIVERY_CARRIER_INVALID,"Delivery Carrier ID invalid must be
numeric 1 - 10."},
    {ERR_DELIVERY_CARRIER_ID_RANGE,"Delivery Carrier ID out of range
must be 1 - 10."},
    {ERR_PAYMENT_MISSING_CLT_KEY,"Payment missing Customer Last Name key
\"CLT*\"."},
    {0,""}
    };
    static char szNoMsg[] = "";
    char*szForm;
    if ( !szMsg  )
    szMsg = szNoMsg;
    if ( iTermId > 0 && IsValidTermId(iTermId) )
    szForm = Term.pClientData[iTermId].szBuffer; //if termid valid use
common terminal static buffer.
    else
    szForm = Term.pClientData[0].szBuffer;//else term id invalid so use
common terminal static buffer.
    switch(iErrorType)
    {
    case ERR_TYPE_WEBDLL:
    for(i=0; errorMsgs[i].szMsg[0]; i++)
    {
    if ( iError == errorMsgs[i].iError )
    break;
    }
    if ( !errorMsgs[i].szMsg[0] )
    i = 1;
    strcpy(szForm,"<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    wsprintf(szForm+strlen(szForm), "Error: TPCCWEB(%d): %s", iError,
errorMsgs[i].szMsg);
    strcat(szForm, "</FORM><BODY></HTML>");
    WriteZString(pECB, szForm);
    break;
```

```
    case ERR_TYPE_SQL:
    strcpy(szForm,"<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    wsprintf(szForm+strlen(szForm), "Error: SQLSVR(%d): %s", iError,
szMsg);
    strcat(szForm, "</FORM><BODY></HTML>");
    WriteZString(pECB, szForm);
    break;
    case ERR_TYPE_DBLIB:
    strcpy(szForm,"<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    wsprintf(szForm+strlen(szForm), "Error: DBLIB(%d): %s", iError,
szMsg);
    strcat(szForm, "</FORM><BODY></HTML>");
    WriteZString(pECB, szForm);
    break;
    case ERR_TYPE_ODBC:
    strcpy(szForm,"<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    wsprintf(szForm+strlen(szForm), "Error: ODBC(%d): %s", iError,
szMsg);
    strcat(szForm, "</FORM><BODY></HTML>");
    WriteZString(pECB, szForm);
    break;
    }
    return;
}
```

error.h

```
#ifndef ERROR_H_INCLUDED
#define ERROR_H_INCLUDED
extern TERM Term;
//error message structure used in ErrorMessage API
typedef struct _SERRORMSG
{
    intiError;//error id of message
    charszMsg[80];//message to sent to browser
} SERRORMSG;
void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr);
void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType, char *szMsg, int iTermId, int iSyncId);
#define ERR_BAD_ITEM_ID1//expected abort record in txnRecord
#define ERR_TYPE_DELIVERY_POST2//expected delivery post failed
#defineERR_TYPE_WEBDLL3//tpcc web generated error
#defineERR_TYPE_SQL4//sql server generated error
#defineERR_TYPE_DBLIB5//dblib generated error
#defineERR_TYPE_ODBC6//odbc generated error
#define ERR_TYPE_SOCKET7//error on communication socket client rte only
#define ERR_TYPE_DEADLOCK8//dblib and odbc only deadlock condition
#defineERR_SUCCESS1000//"Success, no error.
#defineERR_COMMAND_UNDEFINED1001//"Command undefined.
#defineERR_NOT_IMPLEMENTED_YET1002//"Not Implemented Yet.
#defineERR_CANNOT_INIT_TERMINAL1003//"Cannot initialize client
connection.
#defineERR_OUT_OF_MEMORY1004//"insufficient memory.
#defineERR_NEW_ORDER_NOT_PROCESSED1005//"Cannot process new Order form.
#defineERR_PAYMENT_NOT_PROCESSED1006//"Cannot process payment form.
#defineERR_NO_SERVER_SPECIFIED1007//"No Server name specified.
#defineERR_ORDER_STATUS_NOT_PROCESSED1008//"Cannot process order status
form.
#defineERR_W_ID_INVALID1009//"Invalid Warehouse ID.
#defineERR_CAN_NOT_SET_MAX_CONNECTIONS1010//"Insufficient memory to
allocate # connections.
#defineERR_NOSUCH_CUSTOMER1011//"No such customer.
#defineERR_D_ID_INVALID1012//"Invalid District ID Must be 1 to 10.
#defineERR_MAX_CONNECT_PARAM1013//"Max client connections exceeded, run
install to increase.
#defineERR_INVALID_SYNC_CONNECTION1014//"Invalid Terminal Sync ID.
#defineERR_INVALID_TERMID1015//"Invalid Terminal ID.
#defineERR_PAYMENT_INVALID_CUSTOMER1016//"Payment Form, No such Customer.
#defineERR_SQL_OPEN_CONNECTION1017//"SQLOpenConnection API Failed.
#defineERR_STOCKLEVEL_MISSING_THRESHOLD_KEY1018//"Stock Level missing
Threshold key "TT*".
#defineERR_STOCKLEVEL_THRESHOLD_INVALID1019//"Stock Level Threshold
invalid data type range = 1 - 99.
#defineERR_STOCKLEVEL_THRESHOLD_RANGE1020//"Stock Level Threshold out of
range, range must be 1 - 99.
#defineERR_STOCKLEVEL_NOT_PROCESSED1021//"Stock Level not processed.
```

```
#defineERR_NEWORDER_FORM_MISSING_DID1022//"New Order missing District
key "DID*".
#defineERR_NEWORDER_DISTRICT_INVALID1023//"New Order District ID Invalid
range 1 - 10.
#defineERR_NEWORDER_DISTRICT_RANGE1024//"New Order District ID out of
Range. Range = 1 - 10.
#defineERR_NEWORDER_CUSTOMER_KEY1025//"New Order missing Customer key
"CID*".
#defineERR_NEWORDER_CUSTOMER_INVALID1026//"New Order customer id invalid
data type, range = 1 to 3000.
#defineERR_NEWORDER_CUSTOMER_RANGE1027//"New Order customer id out of
range, range = 1 to 3000.
#defineERR_NEWORDER_MISSING_IID_KEY1028//"New Order missing Item Id key
"IID*".
#defineERR_NEWORDER_ITEM_BLANK_LINES1029//"New Order blank order lines
all orders must be continuous.
#defineERR_NEWORDER_ITEMID_INVALID1030//"New Order Item Id is wrong data
type, must be numeric.
#defineERR_NEWORDER_MISSING_SUPPW_KEY1031//"New Order missing Supp_W key
"SP##*".
#defineERR_NEWORDER_SUPPW_INVALID1032//"New Order Supp_W invalid data
type must be numeric.
#defineERR_NEWORDER_MISSING_QTY_KEY1033//"New Order Missing Qty key
"Qty##*".
#defineERR_NEWORDER_QTY_INVALID1034//"New Order Qty invalid must be
numeric range 1 - 99.
#defineERR_NEWORDER_SUPPW_RANGE1035//"New Order Supp_W value out of
range range = 1 - Max Warehouses.
#defineERR_NEWORDER_ITEMID_RANGE1036//"New Order Item Id is out of
range. Range = 1 to 999999.
#defineERR_NEWORDER_QTY_RANGE1037//"New Order Qty is out of range. Range
= 1 to 99.
#defineERR_PAYMENT_DISTRICT_INVALID1038//"Payment District ID is invalid
must be 1 - 10.
#defineERR_NEWORDER_SUPPW_WITHOUT_ITEMID1039//"New Order Supp_W field
entered without a corrisponding Item_Id.
#defineERR_NEWORDER_QTY_WITHOUT_ITEMID1040//"New Order Qty entered
without a corrisponding Item_Id.
#defineERR_NEWORDER_NOITEMS_ENTERED1041//"New Order Blank Items between
items, items must be continuous.
#defineERR_PAYMENT_MISSING_DID_KEY1042//"Payment missing District Key
"DID*".
#defineERR_PAYMENT_DISTRICT_RANGE1043//"Payment District Out of range,
range = 1 - 10.
#defineERR_PAYMENT_MISSING_CID_KEY1044//"Payment missing Customer Key
"CID*".
#defineERR_PAYMENT_CUSTOMER_INVALID1045//"Payment Customer data type
invalid, must be numeric.
#defineERR_PAYMENT_MISSING_CLT1046//"Payment missing Customer Last Name
Key "CLT*".
#defineERR_PAYMENT_LAST_NAME_TO_LONG1047//"Payment Customer last name
longer than 16 characters.
#defineERR_PAYMENT_CUSTOMER_RANGE1048//"Payment Customer ID out of
range, must be 1 to 3000.
#defineERR_PAYMENT_CID_AND_CLT1049//"Payment Customer ID and Last Name
entered must be one or other.
#defineERR_PAYMENT_MISSING_CDI_KEY1050//"Payment missing Customer
district key "CDI*".
#defineERR_PAYMENT_CDI_INVALID1051//"Payment Customer district invalid
must be numeric.
#defineERR_PAYMENT_CDI_RANGE1052//"Payment Customer district out of
range must be 1 - 10.
#defineERR_PAYMENT_MISSING_CWI_KEY1053//"Payment missing Customer
Warehouse key "CWI*".
#defineERR_PAYMENT_CWI_INVALID1054//"Payment Customer Warehouse invalid
must be numeric.
#defineERR_PAYMENT_CWI_RANGE1055//"Payment Customer Warehouse out of
range, 1 to Max Warehouses.
#defineERR_PAYMENT_MISSING_HAM_KEY1056//"Payment missing Amount key
"HAM*".
#defineERR_PAYMENT_HAM_INVALID1057//"Payment Amount invalid data type
must be numeric.
#defineERR_PAYMENT_HAM_RANGE1058//"Payment Amount out of range, 0 -
9999.99.
#defineERR_ORDERSTATUS_MISSING_DID_KEY1059//"Order Status missing
District key "DID*".
#defineERR_ORDERSTATUS_DID_INVALID1060//"Order Status District invalid,
value must be numeric 1 - 10.
#defineERR_ORDERSTATUS_DID_RANGE1061//"Order Status District out of
range must be 1 - 10.
#defineERR_ORDERSTATUS_MISSING_CID_KEY1062//"Order Status missing
Customer key "CID*".
#defineERR_ORDERSTATUS_MISSING_CLT_KEY1063//"Order Status missing
Customer Last Name key "CLT*".
#defineERR_ORDERSTATUS_CLT_RANGE1064//"Order Status Customer last name
longer than 16 characters.
#defineERR_ORDERSTATUS_CID_INVALID1065//"Order Status Customer ID
invalid, range must be numeric 1 - 3000.
#defineERR_ORDERSTATUS_CID_RANGE1066//"Order Status Customer ID out of
range must be 1 - 3000.
#defineERR_ORDERSTATUS_CID_AND_CLT1067//"Order Status Customer ID and
LastName entered must be only one."
#defineERR_DELIVERY_MISSING_OCD_KEY1068//"Delivery missing Carrier ID
key \"OCD*\".
#defineERR_DELIVERY_CARRIER_INVALID1069//"Delivery Carrier ID invalid
must be numeric 1 - 10.
#defineERR_DELIVERY_CARRIER_ID_RANGE1070//"Delivery Carrier ID out of
range must be 1 - 10.
#define ERR_PAYMENT_MISSING_CLT_KEY1071//"Payment missing Customer Last
Name key "CLT*".
#endif
```

getopt.c

```
#ifndef __unix
/* got this off net.sources. */
#include <stdio.h>
#include "getopt.h"
/*
 * get option letter from argument vector
 */
int opterr = 1,/* useless, never set or used */
    optind = 1,/* index into parent argv vector */
    optopt;/* character checked for validity */
char*optarg;/* argument associated with option */
#define BADCH(int)'?'
#define NEEDARG (int)':'
#define EMSG""
getopt(int nargc,char * const * nargv,const char *ostr)
{
    static char*place = EMSG;/* option letter processing */
    register char*oli;/* option letter list index */
    char*strchr();
    if(!*place) {/* update scanning pointer */
    if(optind >= nargc || *(place = nargv[optind]) != '-' || !*++place)
return(EOF);
    if (*place == '-') {/* found "--" */
    ++optind;
    return(EOF);
    }
    }/* option letter okay? */
    if ((optopt = (int)*place++) == (int)':' || !(oli =
strchr(ostr,optopt))) {
    if(!*place) ++optind;
    return (BADCH);
    }
    if (*++oli != ':') {/* don't need argument */
    optarg = NULL;
    if (!*place) ++optind;
    }
    else {/* need an argument */
    if (*place) optarg = place;/* no white space */
    else if (nargc <= ++optind) {/* no arg */
    place = EMSG;
    return(NEEDARG);
    }
     else optarg = nargv[optind];/* white space */
    place = EMSG;
    ++optind;
    }
    return(optopt);/* dump back option letter */
    }
    #endif
```

getopt.h

```
    #ifndef _GETOPT_H_INCLUDED
    #define _GETOPT_H_INCLUDED
    #ifdef __cplusplus
    extern "C" {
    #endif
    extern int optind, optopt;
    extern char *optarg;
    extern int getopt(int argc, char * const *argv, const char *options);
    #ifdef __cplusplus
    }  // end of extern "C"
    #endif
    #endif
```

Httpext.h

```
    /********
    *
    *  Copyright (c) 1995  Process Software Corporation
    *
    *  Copyright (c) 1995  Microsoft Corporation
    *
    *
    *  Module Name  : HttpExt.h
    *
    *  Abstract :
    *
    *     This module contains  the structure definitions and prototypes for
    the
    *     version 1.0 HTTP Server Extension interface.
    *
    *****************/
    #ifndef _HTTPEXT_H_
    #define _HTTPEXT_H_
    #include <windows.h>
    #ifdef __cplusplus
    extern "C" {
    #endif
    #define   HSE_VERSION_MAJOR           1      // major version of this
    spec
    #define   HSE_VERSION_MINOR           0      // minor version of this
    spec
```

```
#define   HSE_LOG_BUFFER_LEN         80
#define   HSE_MAX_EXT_DLL_NAME_LEN  256
typedef   LPVOID  HCONN;
// the following are the status codes returned by the Extension DLL
#define   HSE_STATUS_SUCCESS                          1
#define   HSE_STATUS_SUCCESS_AND_KEEP_CONN            2
#define   HSE_STATUS_PENDING                          3
#define   HSE_STATUS_ERROR                            4
// The following are the values to request services with the
ServerSupportFunction.
//  Values from 0 to 1000 are reserved for future versions of the
interface
#define   HSE_REQ_BASE                                0
#define   HSE_REQ_SEND_URL_REDIRECT_RESP              ( HSE_REQ_BASE + 1 )
#define   HSE_REQ_SEND_URL                            ( HSE_REQ_BASE + 2 )
#define   HSE_REQ_SEND_RESPONSE_HEADER                ( HSE_REQ_BASE + 3 )
#define   HSE_REQ_DONE_WITH_SESSION                   ( HSE_REQ_BASE + 4 )
#define   HSE_REQ_END_RESERVED                        1000
//
//  These are Microsoft specific extensions
//
#define   HSE_REQ_MAP_URL_TO_PATH
(HSE_REQ_END_RESERVED+1)
#define   HSE_REQ_GET_SSPI_INFO
(HSE_REQ_END_RESERVED+2)
//
// passed to GetExtensionVersion
//
typedef struct   _HSE_VERSION_INFO {
    DWORD  dwExtensionVersion;
    CHAR   lpszExtensionDesc[HSE_MAX_EXT_DLL_NAME_LEN];
} HSE_VERSION_INFO, *LPHSE_VERSION_INFO;
//
// passed to extension procedure on a new request
//
typedef struct _EXTENSION_CONTROL_BLOCK {
    DWORD     cbSize;                 // size of this struct.
    DWORD     dwVersion;              // version info of this spec
    HCONN     ConnID;                 // Context number not to be
modified!
    DWORD     dwHttpStatusCode;       // HTTP Status code
    CHAR      lpszLogData[HSE_LOG_BUFFER_LEN];// null terminated log
info specific to this Extension DLL
    LPSTR     lpszMethod;             // REQUEST_METHOD
    LPSTR     lpszQueryString;        // QUERY_STRING
    LPSTR     lpszPathInfo;           // PATH_INFO
    LPSTR     lpszPathTranslated;     // PATH_TRANSLATED
    DWORD     cbTotalBytes;           // Total bytes indicated from
client
    DWORD     cbAvailable;            // Available number of bytes
    LPBYTE    lpbData;                // pointer to cbAvailable bytes
    LPSTR     lpszContentType;        // Content type of client data
    BOOL (WINAPI * GetServerVariable) ( HCONN       hConn,
                                        LPSTR       lpszVariableName,

                                LPVOID      lpvBuffer,
                                        LPDWORD     lpdwSize );
    BOOL (WINAPI * WriteClient)  ( HCONN      ConnID,
                                   LPVOID     Buffer,
                                   LPDWORD    lpdwBytes,
                                   DWORD      dwReserved );
    BOOL (WINAPI * ReadClient)  ( HCONN      ConnID,
                                  LPVOID     lpvBuffer,
                                  LPDWORD    lpdwSize );
    BOOL (WINAPI * ServerSupportFunction)( HCONN      hConn,
                                           DWORD      dwHSERRequest,
                                           LPVOID     lpvBuffer,
                                           LPDWORD    lpdwSize,
                                           LPDWORD    lpdwDataType );
} EXTENSION_CONTROL_BLOCK, *LPEXTENSION_CONTROL_BLOCK;
//
//  these are the prototypes that must be exported from the extension DLL
//
BOOL  WINAPI   GetExtensionVersion( HSE_VERSION_INFO  *pVer );
DWORD WINAPI   HttpExtensionProc(  EXTENSION_CONTROL_BLOCK *pECB );
// the following type declarations is for the server side
typedef BOOL  (WINAPI * PFN_GETEXTENSIONVERSION)( HSE_VERSION_INFO
*pVer );
typedef DWORD (WINAPI * PFN_HTTPEXTENSIONPROC )( EXTENSION_CONTROL_BLOCK
*pECB );
#ifdef __cplusplus
}
#endif
#endif  // end definition _HTTPEXT_H_
```

Install.c

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

```
 *  PURPOSE:Automated installation application for TPC-C Web Kit
 *  Author:Philip Durr
 *  philipdu@Microsoft.com
 */
#include <windows.h>
#include <direct.h>
#include  <io.h>
#include <stdlib.h>
#include <stdio.h>
#include <commctrl.h>
#include "install.h"
HICONhIcon;
HINSTANCEhInst;
DWORDversionExeMS;
DWORDversionExeLS;
DWORDversionExeMM;
DWORDversionDllMS;
DWORDversionDllLS;
staticBOOLbLog;
staticBOOLbConnectionPooling;
staticintiThreads;
staticintiMaxWareHouse;
staticintiDelayMs;
staticintiDeadlockRetry;
staticintiMaxConnections;
staticintiPoolThreadLimit;
staticintiThreadTimeout;
staticintiListenBackLog;
staticintiAcceptExOutstanding;
staticintiDllType;
staticintiMaxPhysicalMemory;//max physical memory in MB
staticintiConnectDelay;
staticcharszVersion[256];
BOOLCALLBACKUpdatedDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
BOOLCALLBACKMainDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
BOOLCALLBACKCopyDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
staticvoidProcessOK(HWND hwnd, char *szDllPath);
staticvoidReadRegistrySettings(void);
staticvoidWriteRegistrySettings(char *szDllPath);
staticintCopyFiles(HWND hDlg, char *szDllPath);
staticBOOLGetInstallPath(char *szDllPath);
staticvoidGetVersionInfo(char *szDLLPath, char *szExePath);
staticBOOLCheckWWWebService(void);
staticBOOLStartWWWebService(void);
staticBOOLStopWWWebService(void);
staticvoidUpdateDialog(HWND hDlg);
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow )
{
    int iRc;
    hInst = hInstance;
    InitCommonControls();
    hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));
    iRc = DialogBox(hInstance, MAKEINTRESOURCE(IDD_DIALOG1),
GetDesktopWindow(), MainDlgProc);
    if ( iRc )
    DialogBoxParam(hInstance, MAKEINTRESOURCE(IDD_DIALOG2),
GetDesktopWindow(), UpdatedDlgProc, (LPARAM)iRc);
    DestroyIcon(hIcon);
    return 0;
}
BOOL CALLBACK UpdatedDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
    switch(uMsg)
    {
    case WM_INITDIALOG:
    switch(lParam)
    {
    case 1:
    SetDlgItemText(hwnd, IDC_RESULTS, "DBLIB TPC-C WEB Client
Installed");
    break;
    case 2:
    SetDlgItemText(hwnd, IDC_RESULTS, "ODBC TPC-C WEB Client Installed");
    break;
    case 3:
    SetDlgItemText(hwnd, IDC_RESULTS, "ODBC Connection Pooling TPC-C WEB
Client Installed");
    break;
    }
    return TRUE;
    case WM_COMMAND:
    if ( wParam == IDOK )
    EndDialog(hwnd, TRUE);
    break;
    default:
    break;
    }
    return FALSE;
}
```

```
BOOL CALLBACK MainDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
    PAINTSTRUCTps;
    MEMORYSTATUSmemoryStatus;
    charszTmp[256];
    static charszDllPath[256];
    static charszExePath[256];
    switch(uMsg)
    {
    case WM_INITDIALOG:
    GlobalMemoryStatus(&memoryStatus);
    iMaxPhysicalMemory= (memoryStatus.dwTotalPhys/ 1048576);
    if ( GetInstallPath(szDllPath) )
    {
    MessageBox(hwnd, "Error internet service inetsrv is not installed.",
NULL, MB_ICONSTOP | MB_OK);
    EndDialog(hwnd, FALSE);
    return TRUE;
    }
    bLog= FALSE;
    iThreads= 4;
    iMaxWareHouse= 500;
    iDelayMs= 500;
    iDeadlockRetry= 3;
    iMaxConnections= 25;
    iPoolThreadLimit= iMaxPhysicalMemory * 2;
    iThreadTimeout= 86400;
    iListenBackLog= 15;
    iAcceptExOutstanding= 40;
    iDllType= IDC_DBLIB;
    bConnectionPooling= FALSE;
    ReadRegistrySettings();
    GetModuleFileName(hInst, szExePath, sizeof(szExePath));
    GetVersionInfo(szDllPath, szExePath);
    if ( bLog )
    CheckDlgButton(hwnd, BN_LOG, 1);
    wsprintf(szTmp, "Version %d.%2.2d.%3.3d", versionExeMS,
versionExeMM, versionExeLS);
    SetDlgItemText(hwnd, IDC_VERSION, szTmp);
    SetDlgItemText(hwnd, IDC_PATH, szDllPath);
    SetDlgItemInt(hwnd, ED_MAXWARE, iMaxWareHouse, FALSE);
    SetDlgItemInt(hwnd, ED_THREADS, iThreads, FALSE);
    SetDlgItemInt(hwnd, ED_MAXCONNECTION, iMaxConnections, FALSE);
    SetDlgItemInt(hwnd, ED_IIS_MAX_THEAD_POOL_LIMIT, iPoolThreadLimit,
FALSE);
    SetDlgItemInt(hwnd, ED_IIS_THREAD_TIMEOUT, iThreadTimeout, FALSE);
    SetDlgItemInt(hwnd, ED_IIS_LISTEN_BACKLOG, iListenBackLog, FALSE);
    SetDlgItemInt(hwnd, ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE,
iAcceptExOutstanding, FALSE);
    SetDlgItemInt(hwnd, ED_USER_CONNECT_DELAY_TIME, iConnectDelay,
FALSE);
    if ( !strcmp(szVersion, "DBLIB") )
    {
    CheckDlgButton(hwnd, IDC_DBLIB, 1);
    CheckDlgButton(hwnd, IDC_ODBC, 0);
    CheckDlgButton(hwnd, IDC_CONNECT_POOL, 0);
    EnableWindow(GetDlgItem(hwnd, IDC_CONNECT_POOL), FALSE);
    EnableWindow(GetDlgItem(hwnd, ED_USER_CONNECT_DELAY_TIME), FALSE);
    }
    else
    {
    CheckDlgButton(hwnd, IDC_DBLIB, 0);
    CheckDlgButton(hwnd, IDC_ODBC, 1);
    CheckDlgButton(hwnd, IDC_CONNECT_POOL, bConnectionPooling);
    EnableWindow(GetDlgItem(hwnd, IDC_CONNECT_POOL), TRUE);
    EnableWindow(GetDlgItem(hwnd, ED_USER_CONNECT_DELAY_TIME), TRUE);
    }
    return TRUE;
    case WM_PAINT:
    if ( IsIconic(hwnd) )
    {
    BeginPaint(hwnd, &ps);
    DrawIcon(ps.hdc, 0, 0, hIcon);
    EndPaint(hwnd, &ps);
    return TRUE;
    }
    break;
    case WM_COMMAND:
    if ( HIWORD(wParam) == BN_CLICKED )
    {
    switch( LOWORD(wParam) )
    {
    case IDC_DBLIB:
    bConnectionPooling = IsDlgButtonChecked(hwnd, IDC_CONNECT_POOL);
    CheckDlgButton(hwnd, IDC_CONNECT_POOL, 0);
    EnableWindow(GetDlgItem(hwnd, IDC_CONNECT_POOL), FALSE);
    EnableWindow(GetDlgItem(hwnd, ED_USER_CONNECT_DELAY_TIME), FALSE);
    return TRUE;
    case IDC_ODBC:
    EnableWindow(GetDlgItem(hwnd, IDC_CONNECT_POOL), TRUE);
    CheckDlgButton(hwnd, IDC_CONNECT_POOL, bConnectionPooling);
    EnableWindow(GetDlgItem(hwnd, ED_USER_CONNECT_DELAY_TIME),
bConnectionPooling);
```

```
    return TRUE;
    case IDC_CONNECT_POOL:
    EnableWindow(GetDlgItem(hwnd, ED_USER_CONNECT_DELAY_TIME),
IsDlgButtonChecked(hwnd, IDC_CONNECT_POOL) );
    return TRUE;
    case IDOK:
    ProcessOK(hwnd, szDllPath);
    return TRUE;
    case IDCANCEL:
    EndDialog(hwnd, FALSE);
    return TRUE;
    default:
    return FALSE;
    }
    }
    break;
    default:
    break;
    }
    return FALSE;
}
static void ProcessOK(HWND hwnd, char *szDllPath)
{
    intd;
    HWNDhDlg;
    intrc;
    if ( IsDlgButtonChecked(hwnd, BN_LOG) )
    bLog = TRUE;
    else
    bLog = FALSE;
    iThreads  = GetDlgItemInt(hwnd, ED_THREADS, &d, FALSE);
    iMaxWareHouse = GetDlgItemInt(hwnd, ED_MAXWARE, &d, FALSE);
    iMaxConnections = GetDlgItemInt(hwnd, ED_MAXCONNECTION, &d, FALSE);
    iPoolThreadLimit = GetDlgItemInt(hwnd, ED_IIS_MAX_THEAD_POOL_LIMIT,
&d, FALSE);
    iThreadTimeout = GetDlgItemInt(hwnd, ED_IIS_THREAD_TIMEOUT,&d,
FALSE);
    iListenBackLog = GetDlgItemInt(hwnd, ED_IIS_LISTEN_BACKLOG, &d,
FALSE);
    iAcceptExOutstanding = GetDlgItemInt(hwnd,
ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE, &d, FALSE);
    if ( IsDlgButtonChecked(hwnd, IDC_DBLIB) )
    iDllType = IDC_DBLIB;
    if ( IsDlgButtonChecked(hwnd, IDC_ODBC) )
    iDllType = IDC_ODBC;
    if ( IsDlgButtonChecked(hwnd, IDC_CONNECT_POOL) )
    bConnectionPooling = TRUE;
    else
    bConnectionPooling = FALSE;
    iConnectDelay = GetDlgItemInt(hwnd, ED_USER_CONNECT_DELAY_TIME, &d,
FALSE);
    ShowWindow(hwnd, SW_HIDE);
    hDlg = CreateDialog(hInst, MAKEINTRESOURCE(IDD_DIALOG3), hwnd,
CopyDlgProc);
    ShowWindow(hDlg, SW_SHOWNA);
    UpdateDialog(hDlg);
    rc = CopyFiles(hDlg, szDllPath);
    if ( !rc )
    {
    ShowWindow(hwnd, SW_SHOWNA);
    DestroyWindow(hDlg);
    MessageBox(hwnd, "Error(s) occured when creating tpcc.dll", NULL,
MB_ICONSTOP | MB_OK);
    EndDialog(hwnd, 0);
    return;
    }
    SetDlgItemText(hDlg, IDC_STATUS, "Updating Registry.");
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    if ( iDllType == IDC_DBLIB )
    {
    strcpy(szVersion, "DBLIB");
    rc = 1;
    }
    else if (!bConnectionPooling )
    {
    strcpy(szVersion, "ODBC");
    rc = 2;
    }
    else
    {
    strcpy(szVersion, "ODBC");
    rc = 3;
    }
    WriteRegistrySettings(szDllPath);
    Sleep(100);
    ShowWindow(hwnd, SW_SHOWNA);
    DestroyWindow(hDlg);
    EndDialog(hwnd, rc);
    return;
}
static void ReadRegistrySettings(void)
{
    HKEYhKey;
    DWORDsize;
```

```
    DWORDtype;
    charszTmp[256];
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\TPCC",
0, KEY_READ, &hKey) == ERROR_SUCCESS )
    {
    size = sizeof(szTmp);
    bLog = FALSE;
    if ( RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    if ( !stricmp(szTmp, "ON") )
    bLog = TRUE;
    iThreads = 4;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "NumberOfDeliveryThreads", 0, &type,
szTmp, &size) == ERROR_SUCCESS )
    iThreads = atoi(szTmp);
    if ( iThreads == 0 )
    iThreads = 4;
    iMaxWareHouse = 500;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type, szTmp,
&size) == ERROR_SUCCESS )
    iMaxWareHouse = atoi(szTmp);
    if ( iMaxWareHouse == 0 )
    iMaxWareHouse = 500;
    iDelayMs = 500;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "BackoffDelay", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iDelayMs = atoi(szTmp);
    if ( iDelayMs == 0 )
    iDelayMs = 500;
    iDeadlockRetry = 3;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iDeadlockRetry = atoi(szTmp);
    if ( !iDeadlockRetry )
    iDeadlockRetry = 3;
    iMaxConnections = 25;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iMaxConnections = atoi(szTmp);
    if ( !iMaxConnections )
    iMaxConnections = 25;
    bConnectionPooling = FALSE;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "ConnectionPooling", 0, &type, szTmp,
&size) == ERROR_SUCCESS )
    if ( !stricmp(szTmp, "ON") )
    bConnectionPooling = TRUE;
    iConnectDelay = 500;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "ConnectionPoolRetryTime", 0, &type,
szTmp, &size) == ERROR_SUCCESS )
    iConnectDelay = atoi(szTmp);
    if ( !iConnectDelay )
    iConnectDelay = 500;
    strcpy(szVersion, "DBLIB");
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "LastInstalledVersion", 0, &type, szTmp,
&size) == ERROR_SUCCESS )
    strcpy(szVersion, szTmp);
    if ( strcmp(szVersion, "DBLIB") != 0 && strcmp(szVersion, "ODBC") !=
0 )
    strcpy(szVersion, "DBLIB");
    RegCloseKey(hKey);
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\Inetinfo\\Parameters", 0,
KEY_READ, &hKey) == ERROR_SUCCESS )
    {
    iPoolThreadLimit =  iMaxPhysicalMemory * 2;
    size = sizeof(iPoolThreadLimit);
    if ( RegQueryValueEx(hKey, "PoolThreadLimit", 0, &type, (char
*)&iPoolThreadLimit, &size) == ERROR_SUCCESS )
    if ( !iPoolThreadLimit )
    iPoolThreadLimit =  iMaxPhysicalMemory * 2;
    iThreadTimeout = 86400;
    size = sizeof(iThreadTimeout);
    if ( RegQueryValueEx(hKey, "ThreadTimeout", 0, &type, (char
*)&iThreadTimeout, &size) == ERROR_SUCCESS )
    if ( !iThreadTimeout )
    iThreadTimeout = 86400;
    iListenBackLog = 15;
    size = sizeof(iListenBackLog);
    if ( RegQueryValueEx(hKey, "ListenBackLog", 0, &type, (char
*)&iListenBackLog, &size) == ERROR_SUCCESS )
    if ( !iListenBackLog )
    iListenBackLog = 15;
    }
    RegCloseKey(hKey);
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parameters", 0, KEY_READ,
&hKey) == ERROR_SUCCESS )
    {
    iAcceptExOutstanding = 40;
```

```
    size = sizeof(iAcceptExOutstanding);
    if ( RegQueryValueEx(hKey, "AcceptExOutstanding", 0, &type, (char
*)&iAcceptExOutstanding, &size) == ERROR_SUCCESS )
    if ( !iAcceptExOutstanding )
    iAcceptExOutstanding = 40;
    }

    RegCloseKey(hKey);
    }
    return;
}
static void WriteRegistrySettings(char *szDllPath)
{
    HKEYhKey;
    DWORDdwDisposition;
    charszTmp[256];
    char*ptr;
    intiRc;
    if ( RegCreateKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\TPCC",
0, NULL, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &hKey,
&dwDisposition) == ERROR_SUCCESS )
    {
    strcpy(szTmp, szDllPath);
    ptr = strstr(szTmp, "tpcc");
    if ( ptr )
    *ptr = 0;
    RegSetValueEx(hKey, "PATH", 0, REG_SZ, szTmp, strlen(szTmp));
    if ( bLog )
    RegSetValueEx(hKey, "LOG", 0, REG_SZ, "ON", 2);
    else
    RegSetValueEx(hKey, "LOG", 0, REG_SZ, "OFF", 3);
    itoa(iThreads, szTmp, 10);
    RegSetValueEx(hKey, "NumberOfDeliveryThreads", 0, REG_SZ, szTmp,
strlen(szTmp));
    itoa(iMaxWareHouse, szTmp, 10);
    RegSetValueEx(hKey, "MaximumWarehouses", 0, REG_SZ, szTmp,
strlen(szTmp));
    itoa(iDelayMs, szTmp, 10);
    RegSetValueEx(hKey, "BackoffDelay", 0, REG_SZ, szTmp, strlen(szTmp));
    itoa(iDeadlockRetry, szTmp, 10);
    RegSetValueEx(hKey, "DeadlockRetry", 0, REG_SZ, szTmp,
strlen(szTmp));
    itoa(iMaxConnections, szTmp, 10);
    RegSetValueEx(hKey, "MaxConnections", 0, REG_SZ, szTmp,
strlen(szTmp));
    itoa(iMaxConnections, szTmp, 10);
    RegSetValueEx(hKey, "MaxConnections", 0, REG_SZ, szTmp,
strlen(szTmp));
    if ( bConnectionPooling )
    RegSetValueEx(hKey, "ConnectionPooling", 0, REG_SZ, "ON", 2);
    else
    RegSetValueEx(hKey, "ConnectionPooling", 0, REG_SZ, "OFF", 3);
    itoa(iConnectDelay, szTmp, 10);
    RegSetValueEx(hKey, "ConnectionPoolRetryTime", 0, REG_SZ, szTmp,
strlen(szTmp));
    RegSetValueEx(hKey, "LastInstalledVersion", 0, REG_SZ, szVersion,
strlen(szVersion));
    RegFlushKey(hKey);
    RegCloseKey(hKey);
    }
    if ( (iRc=RegCreateKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\Inetinfo\\Parameters", 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &hKey, &dwDisposition))
== ERROR_SUCCESS )
    {
    RegSetValueEx(hKey, "PoolThreadLimit", 0, REG_DWORD, (char
*)&iPoolThreadLimit, sizeof(iPoolThreadLimit));
    RegSetValueEx(hKey, "ThreadTimeout", 0, REG_DWORD, (char
*)&iThreadTimeout, sizeof(iThreadTimeout));
    RegSetValueEx(hKey, "ListenBackLog", 0, REG_DWORD, (char
*)&iListenBackLog, sizeof(iListenBackLog));
    RegFlushKey(hKey);
    RegCloseKey(hKey);
    }
    if ( (iRc=RegCreateKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parameters", 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &hKey, &dwDisposition))
== ERROR_SUCCESS )
    {
    RegSetValueEx(hKey, "AcceptExOutstanding", 0, REG_DWORD, (char
*)&iAcceptExOutstanding, sizeof(iAcceptExOutstanding));
    RegFlushKey(hKey);
    RegCloseKey(hKey);
    }
    return;
}
BOOL CALLBACK CopyDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
    if ( uMsg == WM_INITDIALOG )
    {
    SendDlgItemMessage(hwnd, IDC_PROGRESS1, PBM_SETRANGE, 0,
MAKELPARAM(0, 8));
    SendDlgItemMessage(hwnd, IDC_PROGRESS1, PBM_SETSTEP, (WPARAM)1, 0);
    return TRUE;
    }
    return FALSE;
```

```
}
static int CopyFiles(HWND hDlg, char *szDllPath)
{
    HGLOBALhDLL;
    HGLOBALhExe;
    HRSRChResInfo;
    BYTE*pSrc;
    HANDLEhFile;
    DWORDdwSize;
    DWORDd;
    charszTmp[256];
    char*ptr;
    BOOLbSvcRunning;
    bSvcRunning = CheckWWWebService();
    if ( bSvcRunning )
    {
    SetDlgItemText(hDlg, IDC_STATUS, "Stopping Web Service.");
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    StopWWWebService();
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    }
    if ( iDllType == IDC_DBLIB )
    hResInfo = FindResource(hInst, MAKEINTRESOURCE(IDR_TPCCDLL1),
"TPCCDLL");
    else // iDllType == IDC_ODBC
    hResInfo = FindResource(hInst, MAKEINTRESOURCE(IDR_TPCCDLL2),
"TPCCDLL");
    SetDlgItemText(hDlg, IDC_STATUS, "Copying Files...");
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    dwSize = SizeofResource(hInst, hResInfo);
    hDLL = LoadResource(hInst, hResInfo );
    pSrc = (BYTE *)LockResource(hDLL);
    remove(szDllPath);
    if ( !(hFile = CreateFile(szDllPath, GENERIC_WRITE, 0, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL)) )
    return 0;
    if ( !WriteFile(hFile, pSrc, dwSize, &d, NULL) )
    return 0;
    CloseHandle(hFile);

    UnlockResource(hDLL);
    FreeResource(hDLL);
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    if ( iDllType == IDC_DBLIB )
    hResInfo = FindResource(hInst, MAKEINTRESOURCE(IDR_DELIVERY1),
"DELIVERY");
    else
    hResInfo = FindResource(hInst, MAKEINTRESOURCE(IDR_DELIVERY2),
"DELIVERY");
    dwSize = SizeofResource(hInst, hResInfo);
    hExe = LoadResource(hInst, hResInfo );
    pSrc = (BYTE *)LockResource(hExe);
    strcpy(szTmp, szDllPath);
    ptr = strstr(szTmp, "tpcc");
    if ( ptr )
    *ptr = 0;
    strcat(szTmp, "delisrv.exe");
    remove(szTmp);
    if ( !(hFile = CreateFile(szTmp, GENERIC_WRITE, 0, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL)) )
    return 0;
    if ( !WriteFile(hFile, pSrc, dwSize, &d, NULL) )
    return 0;
    CloseHandle(hFile);

    UnlockResource(hExe);
    FreeResource(hExe);
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    //if we stopped service restart it.
    if ( bSvcRunning )
    {
    SetDlgItemText(hDlg, IDC_STATUS, "Starting Web Service.");
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    StartWWWebService();
    }
    SendDlgItemMessage(hDlg, IDC_PROGRESS1, PBM_STEPIT, 0, 0);
    UpdateDialog(hDlg);
    return 1;
}
static BOOL GetInstallPath(char *szDllPath)
{
    HKEYhKey;
    BYTEszTmp[256];
    charszKey[256];
    DWORDsize;
    DWORDsv;
    BOOLbRc;
    intlen;
```

```
    char*ptr;
    szDllPath[0] =  0;
    bRc = TRUE;
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parameters\\Virtual Roots",
0, KEY_ALL_ACCESS, &hKey) == ERROR_SUCCESS )
    {
    sv = sizeof(szKey);
    size = sizeof(szTmp);
    if ( RegEnumValue(hKey, 0, szKey, &sv, NULL, NULL, szTmp, &size) ==
ERROR_SUCCESS )
    {
    strcpy(szDllPath, szTmp);
    bRc = FALSE;
    }
    RegCloseKey(hKey);
    }
    if ( (ptr = strchr(szDllPath, ',')) )
    *ptr = 0;
    len = strlen(szDllPath);
    if ( szDllPath[len-1] != '\\' )
    {
    szDllPath[len] = '\\';
    szDllPath[len+1] = 0;
    }
    strcat(szDllPath, "tpcc.dll");
    return bRc;
}
static void GetVersionInfo(char *szDLLPath, char *szExePath)
{
    DWORDd;
    DWORDdwSize;
    DWORDdwBytes;
    char*ptr;
    VS_FIXEDFILEINFO*vs;
    versionDllMS = 0;
    versionDllLS = 0;
    if ( _access(szDLLPath, 00) == 0 )
    {
    dwSize = GetFileVersionInfoSize(szDLLPath, &d);
    if ( dwSize )
    {
    ptr = (char *)malloc(dwSize);
    GetFileVersionInfo(szDLLPath, 0, dwSize, ptr);
    VerQueryValue(ptr, "\\",&vs, &dwBytes);
    versionDllMS = vs->dwProductVersionMS;
    versionDllLS = vs->dwProductVersionLS;
    free(ptr);
    }
    }
    versionExeMS = 0x7FFF;
    versionExeLS = 0x7FFF;
    dwSize = GetFileVersionInfoSize(szExePath, &d);
    if ( dwSize )
    {
    ptr = (char *)malloc(dwSize);
    GetFileVersionInfo(szExePath, 0, dwSize, ptr);
    VerQueryValue(ptr, "\\",&vs, 0, &dwBytes);
    versionExeMS = vs->dwProductVersionMS;
    versionExeLS = LOWORD(vs->dwProductVersionLS);
    versionExeMM = HIWORD(vs->dwProductVersionLS);
    free(ptr);
    }
    return;
}
static BOOL CheckWWWebService(void)
{
    SC_HANDLEschSCManager;
    SC_HANDLEschService;
    SERVICE_STATUSssStatus;
    schSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_ALL_ACCESS);
    schService = OpenService(schSCManager, TEXT("W3SVC"),
SERVICE_ALL_ACCESS);
    if (schService == NULL)
    return FALSE;
    if (! QueryServiceStatus(schService, &ssStatus) )
    goto ServiceNotRunning;
    if ( !ControlService(schService, SERVICE_CONTROL_STOP, &ssStatus) )
    goto ServiceNotRunning;
    //start Service pending, Check the status until the service is
running.
    if (! QueryServiceStatus(schService, &ssStatus) )
    goto ServiceNotRunning;
    CloseServiceHandle(schService);
    return TRUE;
ServiceNotRunning:
    CloseServiceHandle(schService);
    return FALSE;
}
static BOOL StartWWWebService(void)
{
    SC_HANDLEschSCManager;
    SC_HANDLEschService;
    SERVICE_STATUSssStatus;
```

```
    DWORDdwOldCheckPoint;
    schSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_ALL_ACCESS);
    schService = OpenService(schSCManager, TEXT("W3SVC"),
SERVICE_ALL_ACCESS);
    if (schService == NULL)
    return FALSE;
    if (! StartService(schService, 0, NULL) )
    goto StartWWWebErr;
    //start Service pending, Check the status until the service is
running.
    if (! QueryServiceStatus(schService, &ssStatus) )
    goto StartWWWebErr;
    while( ssStatus.dwCurrentState != SERVICE_RUNNING)
    {

    dwOldCheckPoint = ssStatus.dwCheckPoint;//Save the current
checkpoint.
    Sleep(ssStatus.dwWaitHint);//Wait for the specified interval.
    if ( !QueryServiceStatus(schService, &ssStatus) )//Check the status
again.
    break;
    if (dwOldCheckPoint >= ssStatus.dwCheckPoint)//Break if the
checkpoint has not been incremented.
    break;
    }
    if (ssStatus.dwCurrentState == SERVICE_RUNNING)
    goto StartWWWebErr;
    CloseServiceHandle(schService);
    return TRUE;
StartWWWebErr:
    CloseServiceHandle(schService);
    return FALSE;
}
static BOOL StopWWWebService(void)
{
    SC_HANDLEschSCManager;
    SC_HANDLEschService;
    SERVICE_STATUSssStatus;
    DWORDdwOldCheckPoint;
    schSCManager = OpenSCManager(NULL, NULL, SC_MANAGER_ALL_ACCESS);
    schService = OpenService(schSCManager, TEXT("W3SVC"),
SERVICE_ALL_ACCESS);
    if (schService == NULL)
    return FALSE;
    if (! QueryServiceStatus(schService, &ssStatus) )
    goto StopWWWebErr;
    if ( !ControlService(schService, SERVICE_CONTROL_STOP, &ssStatus) )
    goto StopWWWebErr;
    //start Service pending, Check the status until the service is
running.
    if (! QueryServiceStatus(schService, &ssStatus) )
    goto StopWWWebErr;
    while( ssStatus.dwCurrentState == SERVICE_RUNNING)
    {

    dwOldCheckPoint = ssStatus.dwCheckPoint;//Save the current
checkpoint.
    Sleep(ssStatus.dwWaitHint);//Wait for the specified interval.
    if ( !QueryServiceStatus(schService, &ssStatus) )//Check the status
again.
    break;
    if (dwOldCheckPoint >= ssStatus.dwCheckPoint)//Break if the
checkpoint has not been incremented.
    break;
    }
    if (ssStatus.dwCurrentState == SERVICE_RUNNING)
    goto StopWWWebErr;
    CloseServiceHandle(schService);
    return TRUE;
StopWWWebErr:
    CloseServiceHandle(schService);
    return FALSE;
}
static void UpdateDialog(HWND hDlg)
{
    MSG msg;
    UpdateWindow(hDlg);
    while( PeekMessage(&msg, hDlg, 0, 0, PM_REMOVE) )
    {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
    }
    Sleep(250);
    return;
}
```

install.h

```
    //{{NO_DEPENDENCIES}}
    // Microsoft Developer Studio generated include file.
    // Used by install.rc
    //
    #define IDD_DIALOG1                     101
    #define IDI_ICON1                       102
    #define IDR_TPCCDLL1                    103
```

```
#define IDR_TPCCDLL2                    104
#define IDD_DIALOG2                     105
#define IDI_ICON2                       106
#define IDR_DELIVERY1                   107
#define IDD_DIALOG3                     108
#define IDR_DELIVERY2                   109
#define BN_LOG                          1001
#define ED_KEEP                         1002
#define ED_THREADS                      1003
#define ED_THREADS2                     1004
#define ED_MAXWARE                      1006
#define IDC_PATH                        1007
#define IDC_VERSION                     1009
#define IDC_RESULTS                     1010
#define IDC_PROGRESS1                   1011
#define IDC_STATUS                      1012
#define IDC_BUTTON1                     1013
#define ED_MAXCONNECTION                1014
#define ED_IIS_MAX_THEAD_POOL_LIMIT     1015
#define ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE1017
#define ED_IIS_THREAD_TIMEOUT           1018
#define ED_IIS_LISTEN_BACKLOG           1019
#define IDC_DBLIB                       1021
#define IDC_ODBC                        1022
#define IDC_CONNECT_POOL1023
#define ED_USER_CONNECT_DELAY_TIME1024
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        111
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1022
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

install.rc

```
//Microsoft Developer Studio generated resource script.
//
#include "install.h"
#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
////////////////////////////////////////////////////////////////////////////
////
//
// Dialog
//
IDD_DIALOG1 DIALOGEX 0, 0, 218, 250
STYLE DS_MODALFRAME | DS_CENTER | WS_MINIMIZEBOX | WS_POPUP | WS_CAPTION
|
    WS_SYSMENU
CAPTION "TPC-C Web Client Installation Utility"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT        ED_MAXWARE,183,37,21,12,ES_NUMBER,WS_EX_RTLREADING
    CONTROL         "",BN_LOG,"Button",BS_AUTOCHECKBOX | BS_LEFTTEXT |
                    BS_LEFT | BS_VCENTER | WS_TABSTOP,189,51,15,13,
                    WS_EX_STATICEDGE
    EDITTEXT        ED_THREADS,189,65,15,12,ES_NUMBER,WS_EX_RTLREADING
    EDITTEXT
ED_MAXCONNECTION,170,79,34,12,ES_NUMBER,WS_EX_RTLREADING
    EDITTEXT        ED_IIS_MAX_THEAD_POOL_LIMIT,170,93,34,12,ES_NUMBER,
                    WS_EX_RTLREADING
    EDITTEXT        ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE,170,107,34,12,
                    ES_NUMBER,WS_EX_RTLREADING
    EDITTEXT        ED_IIS_THREAD_TIMEOUT,170,121,34,12,ES_NUMBER,
                    WS_EX_RTLREADING
    EDITTEXT        ED_IIS_LISTEN_BACKLOG,170,135,34,12,ES_NUMBER,
                    WS_EX_RTLREADING
    CONTROL         "DBLIB",IDC_DBLIB,"Button",BS_AUTORADIOBUTTON |
                    WS_TABSTOP,162,152,39,12
    CONTROL         "ODBC",IDC_ODBC,"Button",BS_AUTORADIOBUTTON |
WS_TABSTOP,
                    162,167,39,12
    CONTROL         "",IDC_CONNECT_POOL,"Button",BS_AUTOCHECKBOX |
```

```
                    BS_LEFTTEXT | BS_LEFT | BS_VCENTER |
WS_TABSTOP,189,190,
                    15,13,WS_EX_STATICEDGE
    EDITTEXT        ED_USER_CONNECT_DELAY_TIME,170,206,34,12,ES_NUMBER,
                    WS_EX_RTLREADING
    DEFPUSHBUTTON   "OK",IDOK,51,229,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,117,229,50,14
    EDITTEXT        IDC_PATH,42,22,162,13,ES_AUTOHSCROLL | ES_READONLY
    LTEXT           "Max Number of Warehouses:",IDC_STATIC,42,37,115,12,
                    SS_SUNKEN
    LTEXT           "Write HTML To Log file:",IDC_STATIC,42,51,115,12,
                    SS_SUNKEN
    LTEXT           "Number of Delivery
Threads:",IDC_STATIC,42,65,115,12,
                    SS_SUNKEN
    LTEXT           "Max Number of Connections:",IDC_STATIC,42,79,115,12,
                    SS_SUNKEN
    CTEXT           "Version 1.00.001",IDC_VERSION,42,6,162,14,SS_SUNKEN
|
                    WS_BORDER,WS_EX_CLIENTEDGE
    ICON            IDI_ICON1,IDC_STATIC,9,6,21,20,0,WS_EX_CLIENTEDGE
    LTEXT           "IIS Max Thread Pool Limit:",IDC_STATIC,42,93,115,12,
                    SS_SUNKEN
    LTEXT           "Web Service Backlog Queue
Size:",IDC_STATIC,42,107,115,
                    12,SS_SUNKEN
    LTEXT           "IIS Thread
Timeout:",IDC_STATIC,42,121,115,12,SS_SUNKEN
    LTEXT           "IIS Listen
Backlog:",IDC_STATIC,42,135,115,12,SS_SUNKEN
    LTEXT           "Database
Interface:",IDC_STATIC,42,159,115,12,SS_SUNKEN
    GROUPBOX        "",IDC_STATIC,160,146,44,38
    LTEXT           "Use ODBC Connection
Pooling:",IDC_STATIC,42,190,115,12,
                    SS_SUNKEN
    LTEXT           "Connection Pool Retry Delay:",IDC_STATIC,42,206,
                    115,12,SS_SUNKEN
END
IDD_DIALOG2 DIALOGEX 0, 0, 117, 62
STYLE DS_SETFOREGROUND | DS_3DLOOK | DS_CENTER | WS_POPUP | WS_BORDER
EXSTYLE WS_EX_STATICEDGE
FONT 12, "MS Sans Serif", 0, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,33,45,50,9
    CTEXT           "HTML TPC-C Installation
Successfull",IDC_RESULTS,7,22,
                    102,18,0,WS_EX_CLIENTEDGE
    ICON            IDI_ICON2,IDC_STATIC,50,7,18,20,SS_REALSIZEIMAGE,
                    WS_EX_TRANSPARENT
END
IDD_DIALOG3 DIALOG DISCARDABLE  0, 0, 91, 40
STYLE DS_SYSMODAL | DS_MODALFRAME | DS_3DLOOK | DS_CENTER | WS_CAPTION
CAPTION "Installing TPC-C Web Client"
FONT 12, "Arial Black"
BEGIN
    CONTROL
"Progress1",IDC_PROGRESS1,"msctls_progress32",WS_BORDER,
                    7,20,77,13
    CTEXT           "Static",IDC_STATUS,7,7,77,12,SS_SUNKEN
END
/////////////////////////////////////////////////////////////////////////
////
//
// DESIGNINFO
//
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_DIALOG1, DIALOG
    BEGIN
        LEFTMARGIN, 9
        RIGHTMARGIN, 204
        TOPMARGIN, 6
        BOTTOMMARGIN, 232
    END
    IDD_DIALOG2, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 109
        TOPMARGIN, 7
        BOTTOMMARGIN, 54
    END
    IDD_DIALOG3, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 84
        TOPMARGIN, 7
        BOTTOMMARGIN, 33
    END
END
#endif    // APSTUDIO_INVOKED
#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////
////
```

```
//
// TEXTINCLUDE
//
1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Icon
//
// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON1               ICON    DISCARDABLE     "icon1.ico"
IDI_ICON2               ICON    DISCARDABLE     "icon2.ico"
/////////////////////////////////////////////////////////////////////////////
////
//
// TPCCDLL
//
IDR_TPCCDLL1            TPCCDLL DISCARDABLE     "tpcc1.dll"
IDR_TPCCDLL2            TPCCDLL DISCARDABLE     "tpcc2.dll"
#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
////
//
// Version
//
VS_VERSION_INFO VERSIONINFO
 FILEVERSION 0,4,0,0
 PRODUCTVERSION 0,4,0,0
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "CompanyName", "Microsoft\0"
            VALUE "FileDescription", "install\0"
            VALUE "FileVersion", "0, 4, 0, 0\0"
            VALUE "InternalName", "install\0"
            VALUE "LegalCopyright", "Copyright © 1996\0"
            VALUE "OriginalFilename", "install.exe\0"
            VALUE "ProductName", "Microsoft install\0"
            VALUE "ProductVersion", "0, 4, 0, 0\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END
#endif    // !_MAC
/////////////////////////////////////////////////////////////////////////////
////
//
// DELIVERY
//
IDR_DELIVERY1           DELIVERY DISCARDABLE    "delisrv1.exe"
IDR_DELIVERY2           DELIVERY DISCARDABLE    "delisrv2.exe"
#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
////
#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
/////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```

pipe_routines.c

```
#include <windows.h>
#include <stdio.h>
#include "pipe_routines.h"
#include "trans.h"
#include "tpcc.h"
#include "tux.h"
const char *SERVER_PIPE_PATH = "\\\\.\\pipe\\tpcc_pipe.%d";
const char *CLIENT_PIPE_PATH = "\\\\.\\pipe\\tpcc_pipe.%d";
HANDLE
OpenServerPipe(int PipeNumber, int TimeOut)
{
    HANDLE hPipe, hEvent;
    OVERLAPPED overlapped;
    BOOL bSuccess;
    char PipeName[_MAX_PATH];
    SECURITY_ATTRIBUTESsa;
    PSECURITY_DESCRIPTORpSD;
    _snprintf(PipeName, sizeof(PipeName), SERVER_PIPE_PATH, PipeNumber);
#ifdef _DEBUG
    fprintf(stderr, "opening server pipe %s\n", PipeName);
#endif
    hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
    if (hEvent == INVALID_HANDLE_VALUE)
    {
    fprintf(stderr, "OpenServerPipe(%d): Unable to create event
handle\n", PipeNumber);
    return INVALID_HANDLE_VALUE;
    }
    // create a security descriptor that allows anyone to access the
pipe...
    pSD = (PSECURITY_DESCRIPTOR)malloc( SECURITY_DESCRIPTOR_MIN_LENGTH );
    InitializeSecurityDescriptor(pSD, SECURITY_DESCRIPTOR_REVISION);
    SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL, FALSE);
    sa.nLength= sizeof(sa);
    sa.lpSecurityDescriptor= pSD;
    sa.bInheritHandle= TRUE;
    hPipe = CreateNamedPipe(PipeName,
    PIPE_ACCESS_DUPLEX | FILE_FLAG_OVERLAPPED,
    PIPE_TYPE_MESSAGE | PIPE_READMODE_MESSAGE,
    1,
    sizeof(TUX_MSG),
    sizeof(TUX_MSG),
    0,
    &sa);
    if (hPipe == INVALID_HANDLE_VALUE)
    {
    fprintf(stderr, "OpenServerPipe(%d): CreateHamedPipe failed with
error %d\n", PipeNumber, GetLastError());
    CloseHandle(hEvent);
    return INVALID_HANDLE_VALUE;
    }
    overlapped.hEvent = hEvent;
    ConnectNamedPipe(hPipe, &overlapped);
    bSuccess = TRUE; // wish for the best
    switch (GetLastError())
    {
    case ERROR_PIPE_CONNECTED:
    // someone had connected between the create at the connect call - no
biggie
    break;
    case ERROR_IO_PENDING:
    // no one was waiting for us. Set a timeout and wait for them to
    // connect
    switch(WaitForSingleObject(hEvent, TimeOut))
    {
    case WAIT_OBJECT_0:
    // Someone connected within the timeout period.  Continue processing
    break;
    case WAIT_TIMEOUT:
    bSuccess = FALSE;
    break;
    default:
    fprintf(stderr, "OpenServerPipe(%d): waitforsingleobject failed,
error=%d\n", PipeNumber, GetLastError());
    bSuccess = FALSE;
    break;
    }
    break;
    default:
    fprintf(stderr, "OpenServerPipe(%d): connectnamedpipe failed,
error=%d\n", PipeNumber, GetLastError());
    bSuccess = FALSE;
    break;
    }
    CloseHandle(hEvent);
    if (!bSuccess)
    {
    CloseHandle(hPipe);
    hPipe = INVALID_HANDLE_VALUE;
    }
    return hPipe;
}
HANDLE
```

```
OpenClientPipe(int ClientNumber)
{
    char PipeName[_MAX_PATH];
    HANDLE hPipe;
    DWORD DesiredMode = PIPE_READMODE_MESSAGE;
    _snprintf(PipeName, sizeof(PipeName), CLIENT_PIPE_PATH,
ClientNumber);
#ifdef _DEBUG
    fprintf(stderr, "OpenClientPipe begins for client %d\n",
ClientNumber);
#endif
    while (1)
    {
    hPipe = CreateFile(PipeName,
    GENERIC_READ | GENERIC_WRITE,
    FILE_SHARE_READ | FILE_SHARE_WRITE,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    0);
    if (hPipe != INVALID_HANDLE_VALUE)
    break;
    switch(GetLastError())
    {
    case ERROR_FILE_NOT_FOUND:
    // give the server a chance
#ifdef _DEBUG
    fprintf(stderr, "sleeping\n");
#endif
    Sleep(20);
    break;
    default:
    fprintf(stderr, "OpenClientPipe(%d): error in create of %s.  Error =
%d\n", ClientNumber, PipeName, GetLastError());
    return INVALID_HANDLE_VALUE;
    break;
    }
    }
    if (!SetNamedPipeHandleState(hPipe, &DesiredMode, NULL, NULL))
    {
    fprintf(stderr, "OpenClientPipe(%d), SetNamedPipeHandleStated 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, &overlapped, pnRead, TRUE))
    break;
    if (GetLastError() != ERROR_BROKEN_PIPE)
    fprintf(stderr, "ReadPipe: Readfile 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, &overlapped, pnWritten, TRUE))
    break;
    if (GetLastError() != ERROR_BROKEN_PIPE)
    fprintf(stderr, "WritePipe: Writefile 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.
// Used by TPCC.rc
//
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        101
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

sqlroutines.c

```
#include <windows.h>
#include <stdio.h>
#include "util.h"
#include "trans.h"
#include "tpcc.h"
#include "error.h"
#include "sqlroutines.h"
#include "db.h"
int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr,
char *dberrstr, char *oserrstr);
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext);
BOOL SQLDetectDeadlock(DBPROCESS *dbproc);
staticCRITICAL_SECTIONErrorLogCriticalSection;
BOOL SQLThreadAttach(void)
{
    return TRUE;
}
BOOL SQLThreadDetach(void)
{
    return TRUE;
}
BOOL
SQLInit(void)
{
```

```
#ifdef USE_ODBC
    extern HENV henv;
    if ( SQLAllocEnv(&henv) == SQL_ERROR )
    {
    MessageBox(NULL, "Error SQLAllocEnv()", "Init", MB_OK | MB_ICONSTOP);
    return FALSE;
    }

    #if (ODBCVER >= 0x0300)
    if ( bConnectionPooling )
    {
    /* added to make sure we go into connection pooling mode */
    Beep(100,500);
    Beep(1000,500);
    if ( SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION, (PTR)SQL_OV_ODBC3,
SQL_INTEGER) == SQL_ERROR )
    {
    MessageBox(NULL, "Error SQLSetEnvAttr() SQL_ATTR_ODBC_VERSION",
"Init", MB_OK | MB_ICONSTOP);
    return FALSE;
    }
    if ( SQLSetEnvAttr(henv, SQL_ATTR_CONNECTION_POOLING,
(PTR)SQL_CP_ONE_PER_HENV, SQL_INTEGER) == SQL_ERROR )
    {
    MessageBox(NULL, "Error SQLSetEnvAttr()
SQL_ATTR_CONNECTION_POOLING", "Init", MB_OK | MB_ICONSTOP);
    return FALSE;
    }
    }
    #endif
#else
    extern short iMaxConnections;
    dbinit();
    if ( dbgetmaxprocs() < iMaxConnections )
    {
    if ( dbsetmaxprocs(iMaxConnections) == FAIL )
    {
    //set for fail error message when HttpExtensionProc() is called
because
    //at this point we don't have a pECB so no way to show error message.
    iMaxConnections = -1;
    }
    }
    // install error and message handlers
    dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
    dberrhandle((DBERRHANDLE_PROC)err_handler);
#endif
    InitializeCriticalSection(&ErrorLogCriticalSection);
    return TRUE;
}
void
SQLCleanup(void)
{
#ifdef USE_ODBC
    extern HENV henv;
    SQLFreeEnv(henv);
#else
    dbexit();
#endif
    DeleteCriticalSection(&ErrorLogCriticalSection);
}
/* FUNCTION: int err_handler(DBPROCESS *dbproc, int severity, int dberr,
int oserr, char *dberrstr, char *oserrstr)
 *
 * PURPOSE:This function handles DB-Library errors
 *
 * ARGUMENTS:DBPROCESS*dbprocDBPROCESS id pointer
 *  intseverityseverity of error
 *  intdberrerror id
 *  intoserroperating system specific error code
 *  char*dberrstrprintable error description of dberr
 *  char*oserrstrprintable error description of oserr
 *
 * RETURNS:intINT_CONTINUEcontinue if error is SQLETIME else INT_CANCEL
action
 *
 * COMMENTS:None
 *
 */
#ifndef USE_ODBC
    int err_handler(DBPROCESS *dbproc, int severity, int dberr, int
oserr, char *dberrstr, char *oserrstr)
    {
    PECBINFOpEcbInfo;
    EXTENSION_CONTROL_BLOCK*pECB;
    FILE*fp;
    SYSTEMTIMEsystemTime;
    charszTmp[256];
    intiTermId;
    intiSyncId;
    pEcbInfo = NULL;
    if ((dbproc == NULL) || (DBDEAD(dbproc)))
    {
    ErrorMessage(gpECB, -1, ERR_TYPE_DBLIB, "DBPROC is invalid.",
iTermId, iSyncId);
```

```
    return INT_CANCEL;
    }
    if ( !(pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pECB = gpECB;
    iTermId = 0;
    iSyncId = 0;
    }
    else
    {
    pECB = pEcbInfo->pECB;
    iTermId = pEcbInfo->iTermId;
    iSyncId = pEcbInfo->iSyncId;
    }
    if ( pEcbInfo && pEcbInfo->bFailed )
    return INT_CANCEL;
    if ( oserr != DBNOERR )
    {
    ErrorMessage(pECB, oserr, ERR_TYPE_DBLIB, oserrstr, iTermId,
iSyncId);
    if ( pEcbInfo )
    pEcbInfo->bFailed = TRUE;
    GetLocalTime(&systemTime);
    fp = fopen(szErrorLogPath, "ab");
    EnterCriticalSection(&ErrorLogCriticalSection);
    sprintf(szTmp, "Error: DBLIB(%d): %s", oserr, oserrstr);
    fprintf(fp, "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
    systemTime.wYear, systemTime.wMonth, systemTime.wDay,
    systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
    szTmp);
    LeaveCriticalSection(&ErrorLogCriticalSection);
    fclose(fp);
    }
    return INT_CANCEL;
    }
#endif
/* FUNCTION: int msg_handler(DBPROCESS *dbproc, DBINT msgno, int
msgstate, int severity, char *msgtext)
 *
 * PURPOSE:This function handles DB-Library SQL Server error messages
 *
 * ARGUMENTS:DBPROCESS*dbprocDBPROCESS id pointer
 *  DBINTmsgnomessage number
 *  intmsgstatemessage state
 *  intseveritymessage severity
 *  char*msgtextprintable message description
 *
 * RETURNS:intINT_CONTINUEcontinue if error is SQLETIME else INT_CANCEL
action
 *  INT_CANCELcancel operation
 *
 * COMMENTS:This function also sets the dead lock dbproc variable if
necessary.
 *
 */
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext)
{
    PECBINFOpEcbInfo;
    EXTENSION_CONTROL_BLOCK*pECB;
    FILE*fp;
    SYSTEMTIMEsystemTime;
    charszTmp[256];
    intiTermId;
    intiSyncId;
    if ( !(pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pECB = gpECB;
    iTermId = 0;
    iSyncId = 0;
    }
    else
    {
    pECB = pEcbInfo->pECB;
    iTermId = pEcbInfo->iTermId;
    iSyncId = pEcbInfo->iSyncId;
    }
    if ( (msgno == 5701) || (msgno == 2528) || (msgno == 5703) || (msgno
== 6006) )
    return INT_CONTINUE;
    // deadlock message
    if (msgno == 1205)
    {
    // set the deadlock indicator
    if ( pEcbInfo )
    pEcbInfo->bDeadlock = TRUE;
    else
    ErrorMessage(pECB, -1, ERR_TYPE_SQL, "Error, dbgetuserdata returned
NULL.", iTermId, iSyncId);
    return INT_CONTINUE;
    }
    if ( pEcbInfo && pEcbInfo->bFailed )
    return INT_CANCEL;
    if (msgno == 0)
    return INT_CONTINUE;
```

```
    else
    {
    ErrorMessage(pECB, msgno, ERR_TYPE_SQL, msgtext, iTermId, iSyncId);
    if ( pEcbInfo )
    pEcbInfo->bFailed = TRUE;
    GetLocalTime(&systemTime);
    fp = fopen(szErrorLogPath, "ab");
    EnterCriticalSection(&ErrorLogCriticalSection);
    sprintf(szTmp, "Error: SQLSVR(%d): %s", msgno, msgtext);
    fprintf(fp, "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
    systemTime.wYear, systemTime.wMonth, systemTime.wDay,
    systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
    szTmp);
    LeaveCriticalSection(&ErrorLogCriticalSection);
    fclose(fp);
    }
    return INT_CANCEL;
}
/* FUNCTION: BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK*pECB, int
iTermId, int iSyncId, DBPROCESS **dbproc, char *server, char *database,
char *user, char *password, char *app, int *spid, long *pack_size)
 *
 * PURPOSE:This function opens the sql connection for use.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdterminal id of browser
 *  intiSyncIdsync id of browser
 *  DBPROCESS**dbprocpointer to returned DBPROCESS
 *  char*serverSQL server name
 *  char*databaseSQL server database
 *  char*useruser name
 *  char*passworduser password
 *  char*apppointer to returned application array
 *  int*spidpointer to returned spid
 *  long*pack_sizepointer to returned default pack size
 *
 * RETURNS:BOOLFALSEif successfull
 *  TRUEif an error occurs
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId, DBPROCESS **dbproc, char *server, char *database, char
*user, char *password, char *app, int *spid)
    {
    RETCODErc;
    charbuffer[30];
    PECBINFOpEcbInfo;
    *dbproc = (DBPROCESS *)malloc(sizeof(DBPROCESS));
    if ( !*dbproc )
    return TRUE;
    //set pECB data into dbproc
    pEcbInfo = (PECBINFO)malloc(sizeof(ECBINFO));
    pEcbInfo->bDeadlock = FALSE;
    pEcbInfo->pECB= pECB;
    pEcbInfo->iTermId= iTermId;
    pEcbInfo->iSyncId= iSyncId;
    dbsetuserdata(*dbproc, pEcbInfo);
    if ( SQLAllocConnect(henv, &(*dbproc)->hdbc) == SQL_ERROR )
    {
    ODBCError(*dbproc);
    return TRUE;
    }
    if ( SQLSetConnectOption((*dbproc)->hdbc, SQL_PACKET_SIZE, 4096) ==
SQL_ERROR )
    {
    ODBCError(*dbproc);
    return TRUE;
    }

    rc = SQLConnect((*dbproc)->hdbc, server, SQL_NTS, user, SQL_NTS,
password, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
    {
    ODBCError(*dbproc);
    return TRUE;
    }
    rc = SQLAllocStmt((*dbproc)->hdbc, &(*dbproc)->hstmt);
    if (rc == SQL_ERROR)
    {
    ODBCError(*dbproc);
    return TRUE;
    }
    strcpy(buffer, "use tpcc set nocount on set XACT_ABORT ON");
    rc = SQLExecDirect((*dbproc)->hstmt, buffer, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
    {
    ODBCError(*dbproc);
    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)
    {
    ODBCError(*dbproc);
    return TRUE;
    }
    if ( SQLBindCol((*dbproc)->hstmt, 1, SQL_C_SSHORT, &(*dbproc)->spid,
0, NULL) == SQL_ERROR )
    {
    ODBCError(*dbproc);
    return TRUE;
    }

    if ( SQLFetch((*dbproc)->hstmt) == SQL_ERROR )
    {
    ODBCError(*dbproc);
    return TRUE;
    }
    SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);
    if ( bConnectionPooling )
    SQLDisconnect((*dbproc)->hdbc);
    return FALSE;
    }
#else
    BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId, DBPROCESS **dbproc, char *server, char *database, char
*user, char *password, char *app, int *spid)
    {
    LOGINREC*login;
    PECBINFOpEcbInfo;
    //set local msg proc for login record
    //attach pECB record
    //this is necessary as dblib provides no way to pass user data in a
login structure. So until
    //there is an allocated dbproc we need to use a static which means
that the login attempt must
    //be serialized.
    gpECB = pECB;
    login = dblogin();
    if ( !*user )
    DBSETLUSER(login, "sa");
    else
    DBSETLUSER(login, user);
    DBSETLPWD(login,  password);
    DBSETLHOST(login, app);
    DBSETLPACKET(login, (unsigned short)DEFCLPACKSIZE);
    if ((*dbproc = dbopen(login, server )) == NULL)
    return TRUE;
    //set pECB data into dbproc
    pEcbInfo = (PECBINFO)malloc(sizeof(ECBINFO));
    pEcbInfo->bDeadlock = FALSE;
    pEcbInfo->pECB = pECB;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    dbsetuserdata(*dbproc, pEcbInfo);
    // Use the the right database
    dbuse(*dbproc, database);
    dbcmd(*dbproc, "select @@spid");
    dbsqlexec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
    dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *) spid);
    while (dbnextrow(*dbproc) != NO_MORE_ROWS)
    ;
    }
    dbcmd(*dbproc, "set nocount on");
    dbsqlexec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
    while (dbnextrow(*dbproc) != NO_MORE_ROWS)
    ;
    }
    //rollback transaction on abort
    dbcmd(*dbproc, "set XACT_ABORT ON");
    dbsqlexec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
    while (dbnextrow(*dbproc) != NO_MORE_ROWS)
    ;
    }
    return FALSE;
    }
#endif
/* FUNCTION: BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB,
DBPROCESS *dbproc)
 *
 * PURPOSE:This function closes the sql connection.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  DBPROCESS*dbprocpointer to DBPROCESS
 *
 * RETURNS:BOOLFALSEif successfull
 *  TRUEif an error occurs
```

```
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS
*dbproc)
    {
    if ( dbproc )
    {
    SQLFreeStmt(dbproc->hstmt, SQL_DROP);
    SQLDisconnect(dbproc->hdbc);
    SQLFreeConnect(dbproc->hdbc);
    free(dbproc);
    dbproc = NULL;
    }
    return FALSE;
    }
#else
    BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS
*dbproc)
    {
    if (dbclose(dbproc) == FAIL)
    return TRUE;
    return FALSE;
    }
#endif
/* FUNCTION: SQLStockLevel(EXTENSION_CONTROL_BLOCK*pECB, int iTermId,
int iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
 *
 * PURPOSE:This function handles the stock level transaction.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdterminal id of browser
 *  intiSyncIdsync id of browser
 *  DBPROCESS*dbprocconnection db process id
 *  STOCK_LEVEL_DATA*pStockLevelstock level input / output data structure
 *  shortdeadlock_retryretry count if deadlocked
 *
 * RETURNS:BOOLFALSEif successfull
 *  TRUEif deadlocked
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    int SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
    {
    inttryit;
    PECBINFOpEcbInfo;
    //update pECB and bFailed flag
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
#ifdef USE_ODBC
    if ( ReopenConnection(dbproc) )
    return -3;
#endif
    pStockLevel->num_deadlocks = 0;
    for (tryit=0; tryit<deadlock_retry; tryit++)
    {
    BindParameter(dbproc, 1,SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&pStockLevel->w_id, 0);
    BindParameter(dbproc, 2,SQL_C_STINYINT, SQL_TINYINT, 0, 0,
&pStockLevel->d_id, 0);
    BindParameter(dbproc, 3,SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&pStockLevel->thresh_hold, 0);

    if ( !ExecuteStatement(dbproc, "{call tpcc_stocklevel(?,?,?)}") )
    {
    if ( !SQLDetectDeadlock(dbproc) )
    {
    if ( BindColumn(dbproc, 1, SQL_C_SSHORT, &pStockLevel->low_stock, 0)
)
    return TRUE;
    if ( GetResults(dbproc) )
    return TRUE;
    }
    }
    SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
    if ( SQLDetectDeadlock(dbproc) )
    {
    pStockLevel->num_deadlocks++;
    Sleep(10 * tryit);
    }
    else
    {
    strcpy(pStockLevel->execution_status, "Transaction commited.");
```

```
    return FALSE;
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pStockLevel->execution_status, "Hit deadlock max.");
    return TRUE;
    }
#else
    BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
    {
    inttryit;
    RETCODErc;
    charprintbuf[25];
    BYTE*pData;
    PECBINFOpEcbInfo;
    //update pECB and bFailed flag
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    pStockLevel->num_deadlocks = 0;
    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
    if (dbrpcinit(dbproc, "tpcc_stocklevel", 0) == SUCCEED)
    {
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pStockLevel-
>w_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pStockLevel-
>d_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pStockLevel-
>thresh_hold);
    if (dbrpcexec(dbproc) == SUCCEED)
    {
    while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
    {
    if (DBROWS(dbproc))
    {
    while(((rc=dbnextrow(dbproc))!=NO_MORE_ROWS)&&(rc!=FAIL))
    {
    if(pData=dbdata(dbproc, 1))
    pStockLevel->low_stock = *((long *) pData);
    }
    }
    }
    }
    }
    if (SQLDetectDeadlock(dbproc))
    {
    pStockLevel->num_deadlocks++;
    sprintf(printbuf,"deadlock: retry: %d",pStockLevel->num_deadlocks);
    Sleep(10 * tryit);
    }
    else
    {
    strcpy(pStockLevel->execution_status, "Transaction commited.");
    return FALSE;
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pStockLevel->execution_status, "Hit deadlock max. ");
    return TRUE;
    }
#endif
/* FUNCTION: int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId, int iTermId, int iSyncId, DBPROCESS *dbproc,
NEW_ORDER_DATA  *pNewOrder, short deadlock_retry)
 *
 * PURPOSE:This function handles the new order transaction.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdterminal id of browser
 *  intiSyncIdsync id of browser
 *  DBPROCESS*dbprocconnection db process id
 *  NEW_ORDER_DATA*pNewOrderpointer to new order structure for
input/output data
 *  shortdeadlock_retryretry count if deadlocked
 *
 * RETURNS:intTRUEtransaction committed
 *  FALSEitem number not valid
 *  -1deadlock max retry reached
 *
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    int SQLNewOrder(EXTENSION_CONTROL_BLOCK*pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA  *pNewOrder, short
deadlock_retry)
    {
```

```
    inti;
    intj;
    inttryit;
    DBINTcommit_flag;
    charbuffer[255];
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    if ( ReopenConnection(dbproc) )
    return -3;
    pNewOrder->num_deadlocks = 0;
    for (tryit=0; tryit<deadlock_retry; tryit++)
    {
    strcpy(buffer, "{call tpcc_neworder(?,?,?,?,?,");
    for (i=1; i<pNewOrder->o_ol_cnt; i++)
    strcat(buffer, "?,?,?,");
    strcat(buffer, "?,?,?)}");


    BindParameter(dbproc, 1, SQL_C_SSHORT,   SQL_SMALLINT,   0, 0,
&pNewOrder->w_id, 0);
    BindParameter(dbproc, 2, SQL_C_STINYINT, SQL_TINYINT,    0, 0,
&pNewOrder->d_id, 0);
    BindParameter(dbproc, 3, SQL_C_SLONG,    SQL_INTEGER,    0, 0,
&pNewOrder->c_id, 0);
    BindParameter(dbproc, 4, SQL_C_STINYINT, SQL_TINYINT,    0, 0,
&pNewOrder->o_ol_cnt, 0);


    pNewOrder->o_all_local = 1;
    for (j=0; j<pNewOrder->o_ol_cnt; j++)
    {
    if ( pNewOrder->o_all_local && pNewOrder->Ol[j].ol_supply_w_id !=
pNewOrder->w_id )
    pNewOrder->o_all_local = 0;
    }
    BindParameter(dbproc, 5, SQL_C_STINYINT, SQL_TINYINT,    0, 0,
&pNewOrder->o_all_local, 0);
    for (j=0, i=0; i<(pNewOrder->o_ol_cnt * 3); i=i+3, j++)
    {
    BindParameter(dbproc, (UWORD)(i+6), SQL_C_SLONG,  SQL_INTEGER,  0,
0, &pNewOrder->Ol[j].ol_i_id, 0);
    BindParameter(dbproc, (UWORD)(i+7), SQL_C_SSHORT, SQL_SMALLINT, 0,
0, &pNewOrder->Ol[j].ol_supply_w_id, 0);
    BindParameter(dbproc, (UWORD)(i+8), SQL_C_SSHORT, SQL_SMALLINT, 0,
0, &pNewOrder->Ol[j].ol_quantity, 0);
    }
    if ( ExecuteStatement(dbproc, buffer) )
    if ( !SQLDetectDeadlock(dbproc) )
    return -2;
    pNewOrder->total_amount=0;
    for (i = 0; i<pNewOrder->o_ol_cnt; i++)
    {
    if ( BindColumn(dbproc,1, SQL_C_CHAR,   &pNewOrder->Ol[i].ol_i_name,
sizeof(pNewOrder->Ol[i].ol_i_name)) )
    return -2;
    if ( BindColumn(dbproc,2, SQL_C_SSHORT, &pNewOrder->Ol[i].ol_stock,
0) )
    return -2;
    if ( BindColumn(dbproc,3, SQL_C_CHAR,   &pNewOrder-
>Ol[i].ol_brand_generic, sizeof(pNewOrder->Ol[i].ol_brand_generic)) )
    return -2;
    if ( BindColumn(dbproc,4, SQL_C_DOUBLE, &pNewOrder-
>Ol[i].ol_i_price, 0) )
    return -2;
    if ( BindColumn(dbproc,5, SQL_C_DOUBLE, &pNewOrder->Ol[i].ol_amount,
0) )
    return -2;
    if ( GetResults(dbproc) )
    return -2;
    pNewOrder->total_amount = pNewOrder->total_amount + pNewOrder-
>Ol[i].ol_amount;
    if ( !pEcbInfo->bDeadlock )
    {
    if ( MoreResults(dbproc) )
    return -2;
    }
    if ( pEcbInfo->bDeadlock )
    break;
    }


    if ( !SQLDetectDeadlock(dbproc) )
    {
    if ( BindColumn(dbproc, 1, SQL_C_DOUBLE, &pNewOrder->w_tax, 0) )
    return -2;
    if ( BindColumn(dbproc, 2, SQL_C_DOUBLE,    &pNewOrder->d_tax, 0) )
    return -2;
    if ( BindColumn(dbproc, 3, SQL_C_SLONG,     &pNewOrder->o_id, 0) )
    return -2;
    if ( BindColumn(dbproc, 4, SQL_C_CHAR,      &pNewOrder->c_last,
sizeof(pNewOrder->c_last)) )
    return -2;
```

```
    if ( BindColumn(dbproc, 5, SQL_C_DOUBLE,    &pNewOrder->c_discount,
0) )
    return -2;
    if ( BindColumn(dbproc, 6, SQL_C_CHAR,      &pNewOrder->c_credit,
sizeof(pNewOrder->c_credit)) )
    return -2;
    if ( BindColumn(dbproc, 7, SQL_C_TIMESTAMP, &pNewOrder->o_entry_d,
0) )
    return -2;
    if ( BindColumn(dbproc, 8, SQL_C_SLONG,     &commit_flag, 0) )
    return -2;
    if ( GetResults(dbproc) )
    return -2;
    SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
    if ( commit_flag == 1 )
    {
    pNewOrder->total_amount = pNewOrder->total_amount * ((1 + pNewOrder-
>w_tax + pNewOrder->d_tax) * (1 - pNewOrder->c_discount));
    strcpy(pNewOrder->execution_status,"Transaction commited.");
    return TRUE;
    }
    else
    {
    strcpy(pNewOrder->execution_status,"Item number is not valid.");
    return FALSE;
    }
    }
    else
    {
    SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
    pNewOrder->num_deadlocks++;
    Sleep(DEADLOCKWAIT*tryit);
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pNewOrder->execution_status,"Hit deadlock max.  ");
    return -1;
    }
#else
    int SQLNewOrder(EXTENSION_CONTROL_BLOCK*pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA  *pNewOrder, short
deadlock_retry)
    {
    RETCODErc;
    inti;
    DBINTcommit_flag;
    inttryit;
    charprintbuf[25];
    chartmpbuf[30];
    DBDATETIMEdatetime;
    BYTE*pData;
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    pNewOrder->num_deadlocks = 0;
    strcpy(tmpbuf, "tpcc_neworder");
    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
    if (dbrpcinit(dbproc, tmpbuf, 0) == SUCCEED)
    {
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder-
>w_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder-
>d_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pNewOrder-
>c_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder-
>o_ol_cnt);

    pNewOrder->o_all_local = 1;
    for (i = 0; i < pNewOrder->o_ol_cnt; i++)
    {
    if ( pNewOrder->o_all_local && pNewOrder->Ol[i].ol_supply_w_id !=
pNewOrder->w_id )
    pNewOrder->o_all_local = 0;
    }
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder-
>o_all_local);
    for (i = 0; i < pNewOrder->o_ol_cnt; i++)
    {
    dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pNewOrder-
>Ol[i].ol_i_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder-
>Ol[i].ol_supply_w_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder-
>Ol[i].ol_quantity);
    }
    if (dbrpcexec(dbproc) == SUCCEED)
    {
    pNewOrder->total_amount=0;
    // Get resutls from order line
```

```
    for (i = 0; i<pNewOrder->o_ol_cnt; i++)
    {
    if (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
    {
    if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
    {
    while (dbnextrow(dbproc) != NO_MORE_ROWS)
    {
    if(pData=dbdata(dbproc, 1))
    UtilStrCpy(pNewOrder->Ol[i].ol_i_name, pData, dbdatlen(dbproc, 1));
    if(pData=dbdata(dbproc, 2))
    pNewOrder->Ol[i].ol_stock = (*(DBSMALLINT *) pData);
    if(pData=dbdata(dbproc, 3))
    UtilStrCpy(pNewOrder->Ol[i].ol_brand_generic, pData,
dbdatlen(dbproc, 3));
    if(pData=dbdata(dbproc, 4))
    pNewOrder->Ol[i].ol_i_price = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 5))
    pNewOrder->Ol[i].ol_amount = (*(DBFLT8 *) pData);
    pNewOrder->total_amount = pNewOrder->total_amount + pNewOrder-
>Ol[i].ol_amount;
    }
    }
    }
    }
    while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
    {
    if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
    {
    while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
    {
    if(pData=dbdata(dbproc, 1))
    pNewOrder->w_tax = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 2))
    pNewOrder->d_tax = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 3))
    pNewOrder->o_id = (*(DBINT *) pData);
    if(pData=dbdata(dbproc, 4))
    UtilStrCpy(pNewOrder->c_last, pData, dbdatlen(dbproc, 4));
    if(pData=dbdata(dbproc, 5))
    pNewOrder->c_discount = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 6))
    UtilStrCpy(pNewOrder->c_credit, pData, dbdatlen(dbproc, 6));
    if(pData=dbdata(dbproc, 7))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(dbproc, &pNewOrder->o_entry_d, &datetime);
    }
    if(pData=dbdata(dbproc, 8))commit_flag = (*(DBTINYINT *) pData);
       }
    }
        }
    }
    }
    if (SQLDetectDeadlock(dbproc))
    {
    pNewOrder->num_deadlocks++;
    sprintf(printbuf,"deadlock: retry: %d",pNewOrder->num_deadlocks);
    Sleep(DEADLOCKWAIT*tryit);
    }
    else
    {
    if (commit_flag == 1)
    {
    pNewOrder->total_amount = pNewOrder->total_amount * ((1 + pNewOrder-
>w_tax + pNewOrder->d_tax) * (1 - pNewOrder->c_discount));
    strcpy(pNewOrder->execution_status,"Transaction commited.");
    return TRUE;
    }
    else
    {
    strcpy(pNewOrder->execution_status,"Item number is not valid.");
    return FALSE;
    }
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pNewOrder->execution_status,"Hit deadlock max.  ");
    return -1;//"deadlock max retry reached!"
    }
#endif
/* FUNCTION: int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
deadlock_retry)
 *
 * PURPOSE:This function handles the payment transaction.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdterminal id of browser
 *  intiSyncIdsync id of browser
 *  DBPROCESS*dbprocconnection db process id
 *  PAYMENT_DATA*pPaymentpointer to payment input/output data structure
 *  shortdeadlock_retrydeadlock retry count
```

```
 *
 * RETURNS:intTRUEsuccess
 *  -1max deadlocked reached
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short deadlock_retry)
    {
    inttryit;
    charprintbuf[25];
    charbuffer[255];
    BOOLdeadlock_detected;
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    if ( ReopenConnection(dbproc) )
    return -3;
    pPayment->num_deadlocks = 0;

    for (tryit=0; tryit<deadlock_retry; tryit++)
    {
    deadlock_detected = FALSE;

    strcpy(buffer,"{call tpcc_payment(?,?,?,?,?,?");
    if (pPayment->c_id == 0)
    strcat(buffer,",?");
    strcat(buffer,")}");

    BindParameter(dbproc, 1, SQL_C_SSHORT,   SQL_SMALLINT, 0,      0,
&pPayment->w_id,     0);
    BindParameter(dbproc, 2, SQL_C_SSHORT,   SQL_SMALLINT, 0,      0,
&pPayment->c_w_id,   0);
    BindParameter(dbproc, 3, SQL_C_DOUBLE,   SQL_NUMERIC,  6,      2,
&pPayment->h_amount, 0);
    BindParameter(dbproc, 4, SQL_C_STINYINT, SQL_TINYINT,  0,      0,
&pPayment->d_id,     0);
    BindParameter(dbproc, 5, SQL_C_STINYINT, SQL_TINYINT,  0,      0,
&pPayment->c_d_id,   0);
    BindParameter(dbproc, 6, SQL_C_SLONG,    SQL_INTEGER, (UINT)SQL_NTS,
0, &pPayment->c_id,     0);
    if (pPayment->c_id == 0)
    BindParameter(dbproc, 7, SQL_C_CHAR, SQL_CHAR, (UINT)SQL_NTS, 0,
&pPayment->c_last, sizeof(pPayment->c_last));
    if ( ExecuteStatement(dbproc, buffer) )
    if ( !pEcbInfo->bDeadlock )
    return -2;
    if ( !pEcbInfo->bDeadlock )
    {
    if ( BindColumn(dbproc, 1,  SQL_C_SLONG, &pPayment->c_id, 0) )
    return -2;
    if ( BindColumn(dbproc, 2,  SQL_C_CHAR, &pPayment->c_last,
sizeof(pPayment->c_last)) )
    return -2;
    if ( BindColumn(dbproc, 3,  SQL_C_TIMESTAMP, &pPayment->h_date, 0) )
    return -2;
    if ( BindColumn(dbproc, 4,  SQL_C_CHAR, &pPayment->w_street_1,
sizeof(pPayment->w_street_1)) )
    return -2;
    if ( BindColumn(dbproc, 5,  SQL_C_CHAR, &pPayment->w_street_2,
sizeof(pPayment->w_street_2)) )
    return -2;
    if ( BindColumn(dbproc, 6,  SQL_C_CHAR, &pPayment->w_city,
sizeof(pPayment->w_city)) )
    return -2;
    if ( BindColumn(dbproc, 7,  SQL_C_CHAR, &pPayment->w_state,
sizeof(pPayment->w_state)) )
    return -2;
    if ( BindColumn(dbproc, 8,  SQL_C_CHAR, &pPayment->w_zip,
sizeof(pPayment->w_zip)) )
    return -2;
    if ( BindColumn(dbproc, 9,  SQL_C_CHAR, &pPayment->d_street_1,
sizeof(pPayment->d_street_1)) )
    return -2;
    if ( BindColumn(dbproc, 10, SQL_C_CHAR, &pPayment->d_street_2,
sizeof(pPayment->d_street_2)) )
    return -2;
    if ( BindColumn(dbproc, 11, SQL_C_CHAR, &pPayment->d_city,
sizeof(pPayment->d_city)) )
    return -2;
    if ( BindColumn(dbproc, 12, SQL_C_CHAR, &pPayment->d_state,
sizeof(pPayment->d_state)) )
    return -2;
    if ( BindColumn(dbproc, 13, SQL_C_CHAR, &pPayment->d_zip,
sizeof(pPayment->d_zip)) )
    return -2;
    if ( BindColumn(dbproc, 14, SQL_C_CHAR, &pPayment->c_first,
sizeof(pPayment->c_first)) )
    return -2;
```

```
    if ( BindColumn(dbproc, 15, SQL_C_CHAR, &pPayment->c_middle,
sizeof(pPayment->c_middle)) )
    return -2;
    if ( BindColumn(dbproc, 16, SQL_C_CHAR, &pPayment->c_street_1,
sizeof(pPayment->c_street_1)) )
    return -2;
    if ( BindColumn(dbproc, 17, SQL_C_CHAR, &pPayment->c_street_2,
sizeof(pPayment->c_street_2)) )
    return -2;
    if ( BindColumn(dbproc, 18, SQL_C_CHAR, &pPayment->c_city,
sizeof(pPayment->c_city)) )
    return -2;
    if ( BindColumn(dbproc, 19, SQL_C_CHAR, &pPayment->c_state,
sizeof(pPayment->c_state)) )
    return -2;
    if ( BindColumn(dbproc, 20, SQL_C_CHAR, &pPayment->c_zip,
sizeof(pPayment->c_zip)) )
    return -2;
    if ( BindColumn(dbproc, 21, SQL_C_CHAR, &pPayment->c_phone,
sizeof(pPayment->c_phone)) )
    return -2;
    if ( BindColumn(dbproc, 22, SQL_C_TIMESTAMP, &pPayment->c_since, 0) )
    return -2;
    if ( BindColumn(dbproc, 23, SQL_C_CHAR, &pPayment->c_credit,
sizeof(pPayment->c_credit)) )
    return -2;
    if ( BindColumn(dbproc, 24, SQL_C_DOUBLE, &pPayment->c_credit_lim,
0) )
    return -2;
    if ( BindColumn(dbproc, 25, SQL_C_DOUBLE, &pPayment->c_discount, 0) )
    return -2;
    if ( BindColumn(dbproc, 26, SQL_C_DOUBLE, &pPayment->c_balance, 0) )
    return -2;
    if ( BindColumn(dbproc, 27, SQL_C_CHAR, &pPayment->c_data,
sizeof(pPayment->c_data)) )
    return -2;
    if ( GetResults(dbproc) )
    return -2;
    }

    SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
    if ( SQLDetectDeadlock(dbproc) )
    {
    pPayment->num_deadlocks++;
    sprintf(printbuf,"deadlock: retry: %d",pPayment->num_deadlocks);
    Sleep(DEADLOCKWAIT*tryit);
    }
    else
    {
    if ( pPayment->c_id == 0 )
    {
    strcpy(pPayment->execution_status,"Invalid Customer id,name.");
    return 0;
    }
    else
    strcpy(pPayment->execution_status,"Transaction commited.");
    return TRUE;
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pPayment->execution_status,"Hit deadlock max.  ");
    return -1;
    }
#else
    int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short deadlock_retry)
    {
    RETCODErc;
    inttryit;
    charprintbuf[26];
    DBDATETIMEdatetime;
    BYTE*pData;
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    pPayment->num_deadlocks = 0;

    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
    if (dbrpcinit(dbproc, "tpcc_payment", 0) == SUCCEED)
    {
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pPayment-
>w_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pPayment-
>c_w_id);
    dbrpcparam(dbproc, NULL, 0, SQLFLT8, -1, -1, (BYTE *) &pPayment-
>h_amount);
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pPayment-
>d_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pPayment-
>c_d_id);
```

```
    dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pPayment-
>c_id);
    if (pPayment->c_id == 0)
    {
    dbrpcparam(dbproc, NULL, 0, SQLCHAR, -1, strlen(pPayment->c_last),
pPayment->c_last);
    }
    }
    if (dbrpcexec(dbproc) == SUCCEED)
    {
    while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
    {
    if (DBROWS(dbproc) && (dbnumcols(dbproc) == 27))
    {
    while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
    {
    if(pData=dbdata(dbproc, 1))
    pPayment->c_id = *((DBINT *) pData);
    if(pData=dbdata(dbproc, 2))
    UtilStrCpy(pPayment->c_last, pData, dbdatlen(dbproc, 2));
    if(pData=dbdata(dbproc, 3))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(dbproc, &pPayment->h_date, &datetime);
    }
    if(pData=dbdata(dbproc, 4))
    UtilStrCpy(pPayment->w_street_1, pData, dbdatlen(dbproc, 4));
    if(pData=dbdata(dbproc, 5))
    UtilStrCpy(pPayment->w_street_2, pData, dbdatlen(dbproc, 5));
    if(pData=dbdata(dbproc, 6))
    UtilStrCpy(pPayment->w_city, pData, dbdatlen(dbproc, 6));
    if(pData=dbdata(dbproc, 7))
    UtilStrCpy(pPayment->w_state, pData, dbdatlen(dbproc, 7));
    if(pData=dbdata(dbproc, 8))
    UtilStrCpy(pPayment->w_zip, pData, dbdatlen(dbproc, 8));
    if(pData=dbdata(dbproc, 9))
    UtilStrCpy(pPayment->d_street_1, pData, dbdatlen(dbproc, 9));
    if(pData=dbdata(dbproc, 10))
    UtilStrCpy(pPayment->d_street_2, pData, dbdatlen(dbproc, 10));
    if(pData=dbdata(dbproc, 11))
    UtilStrCpy(pPayment->d_city, pData, dbdatlen(dbproc, 11));
    if(pData=dbdata(dbproc, 12))
    UtilStrCpy(pPayment->d_state, pData, dbdatlen(dbproc, 12));
    if(pData=dbdata(dbproc, 13))
    UtilStrCpy(pPayment->d_zip, pData, dbdatlen(dbproc, 13));
    if(pData=dbdata(dbproc, 14))
    UtilStrCpy(pPayment->c_first, pData, dbdatlen(dbproc, 14));
    if(pData=dbdata(dbproc, 15))
    UtilStrCpy(pPayment->c_middle, pData, dbdatlen(dbproc, 15));
    if(pData=dbdata(dbproc, 16))
    UtilStrCpy(pPayment->c_street_1, pData, dbdatlen(dbproc, 16));
    if(pData=dbdata(dbproc, 17))
    UtilStrCpy(pPayment->c_street_2, pData, dbdatlen(dbproc, 17));
    if(pData=dbdata(dbproc, 18))
    UtilStrCpy(pPayment->c_city, pData, dbdatlen(dbproc, 18));
    if(pData=dbdata(dbproc, 19))
    UtilStrCpy(pPayment->c_state, pData, dbdatlen(dbproc, 19));
    if(pData=dbdata(dbproc, 20))
    UtilStrCpy(pPayment->c_zip, pData, dbdatlen(dbproc, 20));
    if(pData=dbdata(dbproc, 21))
    UtilStrCpy(pPayment->c_phone, pData, dbdatlen(dbproc, 21));
    if(pData=dbdata(dbproc, 22))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(dbproc, &pPayment->c_since, &datetime);
    }
    if(pData=dbdata(dbproc, 23))
    UtilStrCpy(pPayment->c_credit, pData, dbdatlen(dbproc, 23));
    if(pData=dbdata(dbproc, 24))
    pPayment->c_credit_lim = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 25))
    pPayment->c_discount = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 26))
    pPayment->c_balance = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 27))
    UtilStrCpy(pPayment->c_data, pData, dbdatlen(dbproc, 27));
    }
    }
    }
    }
    if (SQLDetectDeadlock(dbproc))
    {
    pPayment->num_deadlocks++;
    sprintf(printbuf,"deadlock: retry: %d",pPayment->num_deadlocks);
    Sleep(DEADLOCKWAIT*tryit);
    }
    else
    {
    if ( pPayment->c_id == 0 )
    {
    strcpy(pPayment->execution_status,"Invalid Customer id,name.");
    return 0;
    }
    else
```

```
    strcpy(pPayment->execution_status,"Transaction commited.");
    return TRUE;
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pPayment->execution_status,"Hit deadlock max.  ");
    return -1; //"deadlock max retry reached!"
    }
#endif
/* FUNCTION: int (EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
 *
 * PURPOSE:This function processes the Order Status transaction.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdterminal id of browser
 *  intiSyncIdsync id of browser
 *  DBPROCESS*dbprocconnection db process id
 *  ORDER_STATUS_DATA*pOrderStatuspointer to Order Status data
input/output structure
 *  shortdeadlock_retrydeadlock retry count
 *
 * RETURNS:int-1max deadlock reached
 *   0No orders found for customer
 *   1Transaction successfull
 *
 * COMMENTS:None
 *
 */
#ifdef USE_ODBC
    int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
    {
    int                 tryit;
    int                 i;
    BOOL                not_done;
    char                buffer[255];
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    if ( ReopenConnection(dbproc) )
    return -3;
    pOrderStatus->num_deadlocks = 0;
    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
    pEcbInfo->bDeadlock = FALSE;

    strcpy(buffer,"{call tpcc_orderstatus(?,?,?");
    if (pOrderStatus->c_id == 0)
    strcat(buffer,",?");
    strcat(buffer,")}");
    BindParameter(dbproc, 1, SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&pOrderStatus->w_id, 0);
    BindParameter(dbproc, 2, SQL_C_STINYINT, SQL_TINYINT, 0, 0,
&pOrderStatus->d_id, 0);
    BindParameter(dbproc, 3, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&pOrderStatus->c_id, 0);
    if (pOrderStatus->c_id == 0)
    BindParameter(dbproc, 4, SQL_C_CHAR, SQL_CHAR, (UINT)SQL_NTS, 0,
&pOrderStatus->c_last, sizeof(pOrderStatus->c_last));
    if ( ExecuteStatement(dbproc, buffer) )
    if ( !SQLDetectDeadlock(dbproc) )
    return -2;
    not_done = TRUE;
    i=0;
    while ( not_done && !pEcbInfo->bDeadlock )
    {
    if ( BindColumn(dbproc, 1, SQL_C_SSHORT, &pOrderStatus-
>OlOrderStatusData[i].ol_supply_w_id, 0) )
    return -2;
    if ( BindColumn(dbproc, 2, SQL_C_SLONG, &pOrderStatus-
>OlOrderStatusData[i].ol_i_id, 0) )
    return -2;
    if ( BindColumn(dbproc, 3, SQL_C_SSHORT, &pOrderStatus-
>OlOrderStatusData[i].ol_quantity, 0) )
    return -2;
    if ( BindColumn(dbproc, 4, SQL_C_DOUBLE, &pOrderStatus-
>OlOrderStatusData[i].ol_amount, 0) )
    return -2;
    if ( BindColumn(dbproc, 5, SQL_C_TIMESTAMP, &pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d, 0) )
    return -2;
    switch( SQLFetch(dbproc->hstmt) )
    {
    case SQL_ERROR:
    if ( !pEcbInfo->bDeadlock )
    return -2;
    break;
```

```
    case SQL_NO_DATA_FOUND:
    not_done = FALSE;
    break;
    default:
    i++;
    break;
    }
    }
    pOrderStatus->o_ol_cnt = i;

    if ( i )
    {
    if ( !pEcbInfo->bDeadlock )
    {
    if ( MoreResults(dbproc) )
    {
    if ( !pEcbInfo->bDeadlock )
    return -2;
    }
    else
    {
    if ( !pEcbInfo->bDeadlock )
    {
    if ( BindColumn(dbproc, 1, SQL_C_SLONG, &pOrderStatus->c_id, 0) )
    return -2;
    if ( BindColumn(dbproc, 2, SQL_C_CHAR, &pOrderStatus->c_last,
sizeof(pOrderStatus->c_last)) )
    return -2;
    if ( BindColumn(dbproc, 3, SQL_C_CHAR, &pOrderStatus->c_first,
sizeof(pOrderStatus->c_first)) )
    return -2;
    if ( BindColumn(dbproc, 4, SQL_C_CHAR, &pOrderStatus->c_middle,
sizeof(pOrderStatus->c_middle)) )
    return -2;
    if ( BindColumn(dbproc, 5, SQL_C_TIMESTAMP, &pOrderStatus-
>o_entry_d, 0) )
    return -2;
    if ( BindColumn(dbproc, 6, SQL_C_SSHORT, &pOrderStatus-
>o_carrier_id, 0) )
    return -2;
    if ( BindColumn(dbproc, 7, SQL_C_DOUBLE, &pOrderStatus->c_balance,
0) )
    return -2;
    if ( BindColumn(dbproc, 8, SQL_C_SLONG, &pOrderStatus->o_id, 0) )
    return -2;
    if ( GetResults(dbproc) )
    return -2;
    }
    }
    }
    }
    else
    {
    SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
    return 0; //"No orders found for customer"
    }
    SQLFreeStmt(dbproc->hstmt, SQL_CLOSE);
    if ( pEcbInfo->bDeadlock )
    {
    pOrderStatus->num_deadlocks++;
    Sleep(DEADLOCKWAIT*tryit);
    }
    else
    {
    if (pOrderStatus->c_id == 0 && pOrderStatus->c_last[0] == 0)
    strcpy(pOrderStatus->execution_status,"Invalid Customer id,name.");
    else
    strcpy(pOrderStatus->execution_status,"Transaction commited.");
    return 1;
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pOrderStatus->execution_status,"Hit deadlock max.  ");
    return -1;
    }
#else
    int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
    {
    RETCODErc;
    inttryit;
    inti;
    charprintbuf[25];
    DBDATETIMEdatetime;
    BYTE*pData;
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    pEcbInfo->pECB = pECB;
    pEcbInfo->bFailed = FALSE;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    }
    pOrderStatus->num_deadlocks = 0;
```

```
    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
    if (dbrpcinit(dbproc, "tpcc_orderstatus", 0) == SUCCEED)
    {
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pOrderStatus-
>w_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pOrderStatus-
>d_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pOrderStatus-
>c_id);
    if (pOrderStatus->c_id == 0)
    {
    dbrpcparam(dbproc, NULL, 0, SQLCHAR, -1, strlen(pOrderStatus-
>c_last), pOrderStatus->c_last);
    }
    }
    if (dbrpcexec(dbproc) == SUCCEED)
    {
    while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
    {
    if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
    {
    i=0;
    while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
    {
    if(pData=dbdata(dbproc, 1))
    pOrderStatus->OlOrderStatusData[i].ol_supply_w_id = (*(DBSMALLINT *)
pData);
    if(pData=dbdata(dbproc, 2))
    pOrderStatus->OlOrderStatusData[i].ol_i_id = (*(DBINT *) pData);
    if(pData=dbdata(dbproc, 3))
    pOrderStatus->OlOrderStatusData[i].ol_quantity = (*(DBSMALLINT *)
pData);
    if(pData=dbdata(dbproc, 4))
    pOrderStatus->OlOrderStatusData[i].ol_amount = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 5))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(dbproc, &pOrderStatus-
>OlOrderStatusData[i].ol_delivery_d, &datetime);
    }
    i++;
    }
    pOrderStatus->o_ol_cnt = i;
    }
    else if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
    {
    while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
    {
    if(pData=dbdata(dbproc, 1))
    pOrderStatus->c_id = (*(DBINT *) pData);
    if(pData=dbdata(dbproc, 2))
    UtilStrCpy(pOrderStatus->c_last, pData, dbdatlen(dbproc,2));
    if(pData=dbdata(dbproc, 3))
    UtilStrCpy(pOrderStatus->c_first, pData, dbdatlen(dbproc,3));
    if(pData=dbdata(dbproc, 4))
    UtilStrCpy(pOrderStatus->c_middle,pData,dbdatlen(dbproc,4));
    if(pData=dbdata(dbproc, 5))
    {
    datetime = *((DBDATETIME *) pData);
    dbdatecrack(dbproc, &pOrderStatus->o_entry_d, &datetime);
    }
    if(pData=dbdata(dbproc, 6))
    pOrderStatus->o_carrier_id = (*(DBSMALLINT *) pData);
    if(pData=dbdata(dbproc, 7))
    pOrderStatus->c_balance = (*(DBFLT8 *) pData);
    if(pData=dbdata(dbproc, 8))
    pOrderStatus->o_id = (*(DBINT *) pData);
    }
    }
    if (i==0)
    return 0; //"No orders found for customer"
    }
    }
    if (SQLDetectDeadlock(dbproc))
    {
    pOrderStatus->num_deadlocks++;
    sprintf(printbuf,"deadlock: retry: %d",pOrderStatus->num_deadlocks);
    Sleep(DEADLOCKWAIT*tryit);
    }
    else
    {
    if (pOrderStatus->c_id == 0 && pOrderStatus->c_last[0] == 0)
    strcpy(pOrderStatus->execution_status,"Invalid Customer id,name.");
    else
    strcpy(pOrderStatus->execution_status,"Transaction commited.");
    return 1;
    }
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pOrderStatus->execution_status,"Hit deadlock max.  ");
    return -1; //"deadlock max retry reached!"
    }
#endif
/* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
```

```
 *
 * PURPOSE:This function checks to see if a sql server deadlock
condition exists.
 *
 * ARGUMENTS:DBPROCESS*dbprocconnection db process id to check
 *
 * RETURNS:BOOLFALSEno deadlock detected
 *  TRUEdeadlock condition exists
 *
 * COMMENTS:None
 *
 */
BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
{
    PECBINFOpEcbInfo;
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
    if ( pEcbInfo->bDeadlock )
    {
    pEcbInfo->bDeadlock = FALSE;
    return TRUE;
    }
    }
    return FALSE;
}
#ifdef USE_ODBC
    /* FUNCTION: void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
     *
     * PURPOSE:This function sets a user pointer in a dbproc structure
     *This functionality is not provided in odbc so this function
     *provides it.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *void *uPtrreturned data user pointer
     *
     * RETURNS:none
     *
     * COMMENTS:The caller is responsible for the contents of the uPtr.
     *
     */
    void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
    {
    dbproc->uPtr = uPtr;
    }
    /* FUNCTION: void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
     *
     * PURPOSE:This function returns the user pointer stored in a dbproc
structure
     *This functionality is not provided in odbc so this function
     *provides it.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:none
     *
     * COMMENTS:The returned pointer is placed in the dbproc structure
by the dbsetuserdata() API.
     *
     */
    void *dbgetuserdata(PDBPROCESS dbproc)
    {
    return dbproc->uPtr;
    }
    /* FUNCTION: void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD
fCType, SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue,
SDWORD cbValueMax)
     *
     * PURPOSE:This function wraps the functionality provided by the
SQLBindParameter
     *allowing error process so that each bind call does not need to
provide
     *error and message checking.
     *
     * ARGUMENTS:PDBPROCESSdbprocpointer to odbc dbprocess structure
     *UWORDiparParameter number, ordered sequentially left to right,
starting at 1.
     *SWORDfParamTypeThe type of the parameter.
     *SWORDfCTypeThe C data type of the parameter.
     *SWORDfSqlTypeThe SQL data type of the parameter.
     *UDWORDcbColDefThe precision of the column or expression
     *of the corresponding parameter marker.
     *SWORDibScaleThe scale of the column or expression of the
corresponding
     *parameter marker.
     *PTRrgbValueA pointer to a buffer for the parameter's data.
     *SDWORDcbValueMaxMaximum length of the rgbValue buffer.
     *void *uPtrreturned data user pointer
     *
     * RETURNS:none
     *
     * COMMENTS:The returned pointer is placed in the dbproc structure
by the dbset
     *
     */
```

```
    void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD fCType,
SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue, SDWORD
cbValueMax)
    {
    RETCODE rc;
    if ( ((PECBINFO)dbgetuserdata(dbproc))->bFailed )
    return;
    rc = SQLBindParameter(dbproc->hstmt, ipar, SQL_PARAM_INPUT, fCType,
fSqlType, cbColDef, ibScale, rgbValue, cbValueMax, NULL);
    if (rc == SQL_ERROR)
    ODBCError(dbproc);
    return;
    }
    /* FUNCTION: void ODBCError(PDBPROCESS dbproc)
     *
     * PURPOSE:This function wraps the odbc error call so that the dblib
msg_handler is called.
     *This allows the deadlock flag in the dbproc user data structure
pEcbInfo in
     *dbproc to be set if necessary.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    void ODBCError(PDBPROCESS dbproc)
    {
    SDWORDlNativeError;
    charszState[6];
    charszMsg[SQL_MAX_MESSAGE_LENGTH];
    charszMsgText[256];
    PECBINFOpEcbInfo;
    charszTmp[256];
    FILE*fp;
    SYSTEMTIMEsystemTime;
    pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
    while( SQLError(henv, dbproc->hdbc, dbproc->hstmt, szState,
&lNativeError, szMsg, sizeof(szMsg), NULL) == SQL_SUCCESS )
    {
    msg_handler(dbproc, lNativeError, 0, 0, szMsg);
    if ( !lNativeError )
    {
    sprintf(szMsgText, "State = %s, %s", szState, szMsg);
    ErrorMessage(pEcbInfo->pECB, -1, ERR_TYPE_ODBC, szMsgText, pEcbInfo-
>iTermId, pEcbInfo->iSyncId);
    pEcbInfo->bFailed = TRUE;
    GetLocalTime(&systemTime);
    fp = fopen(szErrorLogPath, "ab");
    EnterCriticalSection(&ErrorLogCriticalSection);
    sprintf(szTmp, "Error: SQLSVR(): %s", szMsg);
    fprintf(fp, "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
    systemTime.wYear, systemTime.wMonth, systemTime.wDay,
    systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
    szTmp);
    LeaveCriticalSection(&ErrorLogCriticalSection);
    fclose(fp);
    }
    }
    return;
    }
    /* FUNCTION: BOOL ExecuteStatement(PDBPROCESS dbproc, szStatement)
     *
     * PURPOSE:This function wraps the odbc SQLExecDirect API so that
error handling and
     *and deadlock are taken care of in a common location.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *char*szStatementsql stored procedure statement to be executed.
     *
     * RETURNS:none
     *
     * COMMENTS:none
     *
     */
    BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement)
    {
    RETCODErc;
    PECBINFOpEcbInfo;
    pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
    if ( pEcbInfo->bFailed )
    return TRUE;
    rc = SQLExecDirect(dbproc->hstmt, szStatement, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
    {
    ODBCError(dbproc);
    if ( pEcbInfo->bDeadlock )
    return FALSE;
    return TRUE;
    }
    return FALSE;
    }
```

```
   /* FUNCTION: BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol,
SQLSMALLINT fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax)
    *
    * PURPOSE:This function wraps the odbc SQLBindCol API so that error
handling and
    *and deadlock are taken care of in a common location.
    *
    * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
    *UWORDicolColumn number of result data, ordered sequentially left
to right, starting at 1.
    *SWORDfCTypeThe C data type of the result data. SQL_C_BINARY,
SQL_C_BIT, SQL_C_BOOKMARK,
    *SQL_C_CHAR, SQL_C_DATE, SQL_C_DEFAULT, SQL_C_DOUBLE, SQL_C_FLOAT,
SQL_C_SLONG,
    *SQL_C_SSHORT, SQL_C_STINYINT, SQL_C_TIME, SQL_C_TIMESTAMP,
SQL_C_ULONG,
    *SQL_C_USHORT, SQL_C_UTINYINT, SQL_C_DEFAULT
    *PTRrgbValuePointer to storage for the data. If rgbValue is a null
pointer, the
    *driver unbinds the column.
    *SDWORDcbValueMaxMaximum length of the rgbValue buffer. For
character data, rgbValue
    *must also include space for the null-termination byte.
    * RETURNS:none
    *
    * COMMENTS:none
    *
    */
   BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol, SQLSMALLINT
fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax)
   {
   RETCODErc;
   PECBINFOpEcbInfo;
   pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
   if ( pEcbInfo->bFailed )
   return TRUE;
   rc = SQLBindCol(dbproc->hstmt, icol, fCType, rgbValue, cbValueMax,
NULL);
   if ( rc == SQL_ERROR )
   {
   ODBCError(dbproc);
   return TRUE;
   }
   return FALSE;
   }
   /* FUNCTION: BOOL GetResults(PDBPROCESS dbproc)
    *
    * PURPOSE:This function wraps the odbc SQLFetch API so that error
handling and
    *and deadlock are taken care of in a common location.
    *
    * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
    *
    * RETURNS:none
    *
    * COMMENTS:none
    *
    */
   BOOL GetResults(PDBPROCESS dbproc)
   {
   PECBINFO pEcbInfo;
   pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
   if ( pEcbInfo->bFailed )
   return TRUE;
   if ( SQLFetch(dbproc->hstmt) == SQL_ERROR )
   {
   ODBCError(dbproc);
   if ( pEcbInfo->bDeadlock )
   return FALSE;
   return TRUE;
   }
   return FALSE;
   }
   /* FUNCTION: BOOL MoreResults(DBPROCESS  dbproc)
    *
    * PURPOSE:This function wraps the odbc SQLMoreResults API so that
error handling and
    *and deadlock are taken care of in a common location.
    *
    * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
    *
    * RETURNS:none
    *
    * COMMENTS:none
    *
    */
   BOOL MoreResults(PDBPROCESS  dbproc)
   {
   PECBINFOpEcbInfo;
   pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
   if ( pEcbInfo->bFailed )
   return TRUE;
   if ( SQLMoreResults(dbproc->hstmt) == SQL_ERROR )
   {
   ODBCError(dbproc);
   if ( pEcbInfo->bDeadlock )
```

```
    return FALSE;
    return TRUE;
    }
    return FALSE;
    }
    /* FUNCTION: BOOL ReopenConnection(PDBPROCESS dbproc)
     *
     * PURPOSE:This function is used with connection ODBC pooling to
reissue the
     *close hdbc connection.
     *
     * ARGUMENTS:DBRPOCESSdbprocODBC dbprocess structure
     *
     * RETURNS:FALSE if successfull
     *TRUE if an error occurs
     *
     * COMMENTS:none
     *
     */
    BOOL ReopenConnection(PDBPROCESS dbproc)
    {
    RETCODErc;
    PECBINFOpEcbInfo;
    intiCount;
    FILE*fp;
    SYSTEMTIMEsystemTime;
    if ( !bConnectionPooling )
    return FALSE;
    pEcbInfo = (PECBINFO)dbgetuserdata(dbproc);
    iCount = 0;
    /*  I don't think this is necessary.  ODBC connection pooling should
remember this. - damienl
    if ( SQLSetConnectOption(dbproc->hdbc, SQL_PACKET_SIZE, 4096) ==
SQL_ERROR )
    {
    ODBCError(dbproc);
    return TRUE;
    }
    */

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

    rc = SQLConnect(dbproc->hdbc, szServer, SQL_NTS, szUser, SQL_NTS,
szPassword, SQL_NTS);
    while (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
    {
    Sleep(iConnectDelay); //wait and try again
    iCount++;
    if ( (iCount % 1) == 0)
    {
    fp = fopen(szErrorLogPath, "ab");
    GetLocalTime(&systemTime);

    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",
    systemTime.wYear, systemTime.wMonth, systemTime.wDay,
    systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
    pEcbInfo->iTermId, pEcbInfo->iSyncId, iCount);
    fclose(fp);
    }
    rc = SQLConnect(dbproc->hdbc, szServer, SQL_NTS, szUser, SQL_NTS,
szPassword, SQL_NTS);
    }

    rc = SQLAllocStmt(dbproc->hdbc, &dbproc->hstmt);
    if (rc == SQL_ERROR)
    {
    ODBCError(dbproc);
    return TRUE;
    }
    rc = SQLExecDirect((dbproc)->hstmt, "use tpcc set nocount on set
XACT_ABORT ON", SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
    {
    ODBCError(dbproc);
    return TRUE;
    }
    SQLFreeStmt((dbproc)->hstmt, SQL_CLOSE);

    return FALSE;
    }
#endif
PECBINFO SQLGetECB(PDBPROCESS p)
{
    return (PECBINFO)dbgetuserdata(p);
}
```

sqlroutines.h

```
//this structure allows the EXTENSION CONTROL BLOCK to be passed to the
msg and error handlers.
typedef struct _ECBINFO
{
    intiTermId;//terminal id
    intiSyncId;//browser sync id
    BOOLbDeadlock;//deadlock condition flag
    BOOLbFailed;//cleared before sql transaction, set in err handlers if
an error occurs
    EXTENSION_CONTROL_BLOCK *pECB;//inetsrv current connection structure
information
} ECBINFO, *PECBINFO;
BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS **dbproc, char *server, char *database, char *user,
char *password, char *app, int *spid);
BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS
*dbproc);
BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry);
int SQLNewOrder(EXTENSION_CONTROL_BLOCK*pECB, int iTermId, int iSyncId,
DBPROCESS *dbproc, NEW_ORDER_DATA  *pNewOrder, short deadlock_retry);
int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short deadlock_retry);
int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry);
BOOL SQLInit(void);
void SQLCleanup(void);
BOOL SQLThreadAttach(void);
BOOL SQLThreadDetach(void);
PECBINFO SQLGetECB(PDBPROCESS p);
```

tpcc.c

```
/*  FILE:TPCC.C
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96By Francois Raab
 *
 *  Copyright Microsoft, 1996
 *
 *  PURPOSE:Main module for TPCC.DLL which is an ISAPI service dll.
 *  Author:Philip Durr
 *  philipdu@Microsoft.com
 */
#include <windows.h>
#include <process.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <io.h>
#include <fcntl.h>
#include "trans.h"//tpckit transaction header contains definations of
structures specific to TPC-C
#include "httpext.h"//ISAPI DLL information header
#include "tpcc.h"//this dlls specific structure, value e.t. header.
#include "sqlroutines.h"// the header files for the SQL routines (may be
hiding TUX)
#include "util.h"
#include "error.h"
#ifdef USE_ODBC
    HENVhenv;
#endif
charszServer[32]= { 0 };//global variables used with this DLL
charszUser[32]= { 0 };
charszPassword[32]= { 0 };
charszDatabase[32]= "tpcc";
BOOLbLog= FALSE;
int iThreads= 5;
int iMaxWareHouses= 500;
int iQSlotts= 3000;
int iDelayMs= 100;
int iConnectDelay= 500;
shortiDeadlockRetry= (short)3;
shortiMaxConnections = (short)25;
#ifdef USE_ODBC
    intbConnectionPooling = FALSE;
#endif
//allowable client command strings i.e. CMD=command
char *szCmds[] =
{
    "..NewOrder..", "..Payment..", "..Delivery..", "..Order-Status..",
"..Stock-Level..", "..Exit..",
    "Submit", "Begin", "Process", "Menu", "Clear", "Users", ""
};
//defined command string functions, called via CMD=command http string
from html client.
void (*DoCmd[])(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId) =
{
    NewOrderForm,
    PaymentForm,
```

```
    DeliveryForm,
    OrderStatusForm,
    StockLevelForm,
    Exitcmd,
    SubmitCmd,
    BeginCmd,
    ProcessCmd,
    MenuCmd,
    ClearCmd,
    NumberOfConnectionsCmd
};
//Terminal client id structure and interface defination
TERMTerm = { 0, 0, 0, FALSE, NULL, TermInit, TermAllocate, TermRestore,
TermAdd, TermDelete };
//welcome to tpc-c html form buffer, this is first form client sees.
static char*szWelcomeForm ="<HTML>"
    "<HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY>"
    "Please Identify your Warehouse and District for this session.<BR>"
    "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">"
    "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">"
    "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"1\">"
    "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"-2\">"
    "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"0\">"
    "Warehouse ID <INPUT NAME=\"w_id\" SIZE=4><BR>"
    "District ID <INPUT NAME=\"d_id\" SIZE=2><BR>"
    "<HR>"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Submit\">"
    "</FORM><BODY>"
    "</HTML>";
static charszTpccLogPath[256];//path to html log file if logging turned
on in registry.
char szErrorLogPath[256];//path to error log file.
staticCRITICAL_SECTIONCriticalSection;
staticLPTSTRlpszPipeName= TEXT("\\\\.\\pipe\\DELISRV");
staticHANDLEhDeliveryWrite= INVALID_HANDLE_VALUE;
staticHANDLEhPipe= INVALID_HANDLE_VALUE;
EXTENSION_CONTROL_BLOCK*gpECB;
staticintbTpccExit;//exit delivery disconnect loop as dll exiting.
/* FUNCTION: BOOL APIENTRY DllMain(HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved)
 *
 * PURPOSE:This function is the entry point for the DLL this
implementation is baised on the
 *  fact that DLL_PROCESS_ATTACH is only called from the inet service
once. Connections
 *  are sent to this function as thread attachments.
 *
 * ARGUMENTS:HANDLEhModulemodule handle
 *  DWORDul_reason_for_callreason for call
 *  LPVOIDlpReservedreserved for future use
 *
 * RETURNS:BOOLFALSEerrors occured in initialization
 *  TRUEDLL successfully initialized
 *
 * COMMENTS:None
 *
 */
BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID
lpReserved)
{
    inti;
    static SECURITY_ATTRIBUTESsa;
    static PSECURITY_DESCRIPTOR pSD;
    switch( ul_reason_for_call )
    {
    case DLL_PROCESS_ATTACH:
    {
    freopen("\\temp\\tpcc.log", "a", stderr);
    setbuf(stderr, NULL);
    fprintf(stderr, "logging started\n");
    }
    if ( ReadRegistrySettings() )
    {
    MessageBox(NULL, "Cannot Find TPCC Key in registry (run
install.exe).", "Init", MB_OK | MB_ICONSTOP);
    return FALSE;
    }
    InitializeCriticalSection(&CriticalSection);
    (*Term.Init)();
    if ( !(*Term.Allocate)() )
    {
    MessageBox(NULL, "Error Trm.Allocate().", "Init", MB_OK |
MB_ICONSTOP);
    return FALSE;
    }
    for(i=Term.iNext; i<Term.iAvailable; i++)
    Term.pClientData[i].inUse = 0;
    Term.pClientData[0].inUse = 1;
    // create a security descriptor that allows anyone to access the
pipe...
    pSD = (PSECURITY_DESCRIPTOR)malloc( SECURITY_DESCRIPTOR_MIN_LENGTH );
    if ( pSD == NULL )
    {
    MessageBox(NULL, "Error malloc(SECURITY_DESCRIPTOR_MIN_LENGTH)",
"Init", MB_OK | MB_ICONSTOP);
    return FALSE;
```

```
    }
    if ( !InitializeSecurityDescriptor(pSD,
SECURITY_DESCRIPTOR_REVISION) )
    {
    MessageBox(NULL, "Error InitializeSecurityDescriptor()", "Init",
MB_OK | MB_ICONSTOP);
    return FALSE;
    }
    // add a NULL disc. ACL to the security descriptor.
    if ( !SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL, FALSE) )
    {
    MessageBox(NULL, "Error SetSecurityDescriptorDacl().", "Init", MB_OK
| MB_ICONSTOP);
    return FALSE;
    }
    sa.nLength= sizeof(sa);
    sa.lpSecurityDescriptor= pSD;
    sa.bInheritHandle= TRUE;
    // open delivery named pipe...
    hPipe = CreateNamedPipe(lpszPipeName, FILE_FLAG_OVERLAPPED |
PIPE_ACCESS_DUPLEX,
    PIPE_TYPE_BYTE | PIPE_READMODE_BYTE | PIPE_NOWAIT,
    1, 65535, 65535, 250, &sa);
    if ( hPipe == INVALID_HANDLE_VALUE )
    {
    MessageBox(NULL, "Error CreateNamedPipe().", "Init", MB_OK |
MB_ICONSTOP);
    free(pSD);
    return FALSE;
    }
    bTpccExit = FALSE;
    if ( _beginthread( DeliveryDisconnect, 0, NULL ) == -1 )
    {
    MessageBox(NULL, "Error _beginthread()", "Init", MB_OK |
MB_ICONSTOP);
    return FALSE;
    }
    if (!SQLInit())
    return FALSE;
    break;
    case DLL_THREAD_ATTACH:
    if (!SQLThreadAttach())
    return FALSE;
    break;
    case DLL_THREAD_DETACH:
    if (!SQLThreadDetach())
    return FALSE;
    break;
    case DLL_PROCESS_DETACH:
    if ( pSD )
    free( pSD );
    bTpccExit = TRUE;
    if ( hPipe )
    DisconnectNamedPipe(hPipe);
    if (hPipe != INVALID_HANDLE_VALUE )
    CloseHandle(hPipe);
    (*Term.Restore)();
    SQLCleanup();
    DeleteCriticalSection(&CriticalSection);
    break;
    }
    return TRUE;
}
/* FUNCTION: void DeliveryDisconnect(void *ptr)
 *
 * PURPOSE:This function handles disconnecting the server side of the
delivery pipe when the
 *  delivery handler application shuts down.
 *
 * ARGUMENTS:void*ptrvoid pointer normally NULL passed from thread
handler.
 *
 * RETURNS:None
 *
 * COMMENTS:This function runs as thread which allows the client pipe to
disconnect by
 *  sending a byte back though the pipe to the server i.e. this DLL.
 */
static void DeliveryDisconnect(void *ptr)
{
    intl, d;
    SECURITY_ATTRIBUTESsa;
    PSECURITY_DESCRIPTORpSD;
    // create a security descriptor that allows anyone to access the
pipe...
    pSD = (PSECURITY_DESCRIPTOR)malloc( SECURITY_DESCRIPTOR_MIN_LENGTH );
    InitializeSecurityDescriptor(pSD, SECURITY_DESCRIPTOR_REVISION);
    SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL, FALSE);
    sa.nLength= sizeof(sa);
    sa.lpSecurityDescriptor= pSD;
    sa.bInheritHandle= TRUE;
    while( !bTpccExit )
    {
    if ( hPipe && ReadFile(hPipe, &l, 1, &d, NULL) )
    {
```

```
    DisconnectNamedPipe(hPipe);
    CloseHandle(hPipe);
    // open delivery named pipe...
    hPipe = CreateNamedPipe(lpszPipeName, FILE_FLAG_OVERLAPPED |
PIPE_ACCESS_DUPLEX,
    PIPE_TYPE_BYTE | PIPE_READMODE_BYTE | PIPE_NOWAIT,
    1, 65535, 65535, 250, &sa);
    }
    Sleep( 2000 );//check for delivery application exit once every 2
seconds.
    }
    free(pSD);
    return;
}
/* FUNCTION: BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
 *
 * PURPOSE:This function is called by the inet service when the DLL is
first loaded.
 *
 * ARGUMENTS:HSE_VERSION_INFO*pVerpassed in structure in which to place
expected version number.
 *
 * RETURNS:TRUEinet service expected return value.
 *
 * COMMENTS:None
 *
 */
BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
{
    pVer->dwExtensionVersion = MAKELONG(HSE_VERSION_MINOR,
HSE_VERSION_MAJOR);
    lstrcpyn(pVer->lpszExtensionDesc, "TPC-C Server.",
HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
}
/* FUNCTION: DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK
*pECB)
 *
 * PURPOSE:This function is the main entry point for the TPCC DLL. The
internet service
 *  calls this function passing in the http string.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in
internet
 *  service information.
 *
 * RETURNS:DWORDHSE_STATUS_SUCCESSconnection can be dropped if error
 *  HSE_STATUS_SUCCESS_AND_KEEP_CONNkeep connect valid comment sent
 *
 * COMMENTS:None
 *
 */
DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
    int iCmd, FormId, TermId, iSyncId;
    FILE *fp;
    if ( iMaxConnections == -1 )
    {
    ErrorMessage(pECB, ERR_CAN_NOT_SET_MAX_CONNECTIONS, ERR_TYPE_WEBDLL,
NULL, -1, -1);
    return HSE_STATUS_SUCCESS;
    }
    //if registry setting is for html logging then show http string
passed in.
    if ( bLog )
    {
    SYSTEMTIMEsystemTime;
    fp = fopen(szTpccLogPath, "ab");
    GetLocalTime(&systemTime);
    fprintf(fp, "* QUERY * %2.2d/%2.2d/%2.2d
%2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
    systemTime.wYear, systemTime.wMonth, systemTime.wDay,
    systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
    pECB->lpszQueryString);
    fclose(fp);
    }
    //process http query
    if ( !ProcessQueryString(pECB, &iCmd, &FormId, &TermId, &iSyncId) )
    {
    if ( TermId < 0 )
    ErrorMessage(pECB, ERR_INVALID_TERMID, ERR_TYPE_WEBDLL, NULL,
TermId, iSyncId);
    else
    ErrorMessage(pECB, ERR_COMMAND_UNDEFINED, ERR_TYPE_WEBDLL, NULL,
TermId, iSyncId);
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    if ( TermId != 0 )
    {
    if ( !IsValidTermId(TermId) )
    {
    ErrorMessage(pECB, ERR_INVALID_TERMID, ERR_TYPE_WEBDLL, NULL,
TermId, iSyncId);
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    //must have a valid syncid here since termid is valid
```

```
    if ( iSyncId < 1 || iSyncId != Term.pClientData[TermId].iSyncId )
    {
    ErrorMessage(pECB, ERR_INVALID_SYNC_CONNECTION, ERR_TYPE_WEBDLL,
NULL, TermId, iSyncId);
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    }
    //set use time
    Term.pClientData[TermId].iTickCount = GetTickCount();
    //go execute http: command
    (*DoCmd[iCmd])(pECB, FormId, TermId, iSyncId);
    //finish up and keep connection
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}
/* FUNCTION: static BOOL IsValidTermId(int TermId)
 *
 * PURPOSE:This function checks to see of the passed in terminal id is
valid.
 *
 * ARGUMENTS:intTermIdclient terminal id
 *
 * RETURNS:BOOLFALSETerminal ID Invalid
 *  TRUETerminal ID valid
 *
 * COMMENTS:None
 *
 */
BOOL IsValidTermId(int TermId)
{
    return (BOOL) ( TermId > 0 && TermId <= Term.iAvailable &&
Term.pClientData[TermId].inUse );
}
/* FUNCTION: BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int
*pCmd, int *pFormId, int *pTermId, int *pSyncId)
 *
 * PURPOSE:This function extracts the relevent information out of the
http command passed in from
 *  the browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in
internet
 *  service information.
 *  int*pCmdreturned command id
 *  int*pFormIdreturned active form client browser is on
 *  int*pTermIdreturned client terminal id
 *
 * RETURNS:BOOLFALSEsuccess
 *  TRUEcommand passed in is invalid
 *
 * COMMENTS:If this is the initial connection i.e. client is at welcome
screen then
 *  there will not be a terminal id or current form id if this is the
case
 *  then the pTermid and pFormid return values are undefined.
 */
BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd, int
*pFormId, int *pTermId, int *pSyncId)
{
    char *ptr;
    char szBuffer[25];
    char szTmp[25];
    char *dest = szBuffer;
    int i;
    if ( (ptr = strstr(pECB->lpszQueryString, "FORMID=")) )
    *pFormId = *(ptr+7) & 0x0F;
    if ( (ptr = strstr(pECB->lpszQueryString, "TERMID=")) )
    {
    *pTermId = atoi((ptr+7));
    if ( *pTermId == 0 )//terminal id 0 used internally
    *pTermId = -1;
    if ( *pTermId == -2 )//login screen
    *pTermId = 0;
    }
    else
    *pTermId = 0;
    if ( (ptr = strstr(pECB->lpszQueryString, "SYNCID=")) )
    *pSyncId = atoi((ptr+7));
    else
    *pSyncId = 0;
    if ( !(ptr = strstr(pECB->lpszQueryString, "CMD=")) )
    {
    ptr = szBuffer;
    if ( !stricmp(szBuffer, "Default") )
    strcpy(szBuffer, "CMD=Begin");
    switch( *pFormId )
    {
    case WELCOME_FORM:
    strcpy(szBuffer, "CMD=Submit");
    break;
    case MAIN_MENU_FORM:
    strcpy(szBuffer, "CMD=NewOrder");
    break;
    case NEW_ORDER_FORM:
    case PAYMENT_FORM:
    case DELIVERY_FORM:
```

```
    case ORDER_STATUS_FORM:
    case STOCK_LEVEL_FORM:
    if ( !(*pTermId) )
    return FALSE;
    if ( GetKeyValue(pECB->lpszQueryString, "PI*", szTmp, sizeof(szTmp))
)
    strcpy(szBuffer, "CMD=Process");
    else
    {
    strcpy(szBuffer, "CMD=");
    strcat(szBuffer, szCmds[*pFormId - NEW_ORDER_FORM]);
    }
    break;
    default:
    return FALSE;
    }
    }
    ptr += 4;

    while( *ptr && *ptr != '&' )
    *dest++ = *ptr++;
    *dest = 0;
    for(i=0; szCmds[i][0]; i++)
    {
    if ( !strcmp(szCmds[i], szBuffer) )
    {
    *pCmd = i;
    return TRUE;
    }
    }
    return FALSE;
}
/* FUNCTION: void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:This function wraps the functionality needed for the TPC-C
New Order Form.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, TRUE, FALSE));
    UNUSEDPARAM(iFormId);
    return;
}
/* FUNCTION: void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:This function wraps the functionality needed for the TPC-C
Payment Form.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakePaymentForm(iTermId, iSyncId, TRUE) );
    UNUSEDPARAM(iFormId);
}
/* FUNCTION: void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:This function wraps the functionality needed for the TPC-C
Delivery Form.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
```

```
{
    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, TRUE, TRUE) );
    UNUSEDPARAM(iFormId);
}
/* FUNCTION: void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:This function wraps the functionality needed for the TPC-C
Order Status Form.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling borwser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakeOrderStatusForm(iTermId, iSyncId, TRUE) );
    UNUSEDPARAM(iFormId);
}
/* FUNCTION: void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:This function wraps the functionality needed for the TPC-C
Stock Level Form.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakeStockLevelForm(iTermId, iSyncId, TRUE) );
    return;
}
/* FUNCTION: void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function removes a terminal id from use, the allocated
structure however remains
 *  valid so the next request for a new client will not require a new
memory allocation.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    (*Term.Delete)(pECB, iTermId);
    WriteZString(pECB, MakeWelcomeForm() );
    UNUSEDPARAM(iFormId);
    UNUSEDPARAM(iSyncId);
    return;
}
/* FUNCTION: void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function allocated a new terminal id in the Term
structure array.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:A terminal id can be allocated but still be invalid if the
requested warehouse number
 *  is outside the range specified in the registry. This then will force
the client id
 *  to be invalid and an error message sent to the users browser.
 */
void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
```

```
{
    intiCurrent;
    if ( (iCurrent = (*Term.Add)(pECB, pECB->lpszQueryString)) < 0 )
    {
    ErrorMessage(pECB, ERR_CANNOT_INIT_TERMINAL, ERR_TYPE_WEBDLL, NULL,
iCurrent, iSyncId);
    return;
    }
    if ( Term.pClientData[iCurrent].w_id > iMaxWareHouses ||
Term.pClientData[iCurrent].w_id < 1 )
    {
    ErrorMessage(pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL, NULL,
iCurrent, iSyncId);
    (*Term.Delete)(pECB, iCurrent);
    return;
    }
    if ( Term.pClientData[iCurrent].d_id < 1 ||
Term.pClientData[iCurrent].d_id > 10 )
    {
    ErrorMessage(pECB, ERR_D_ID_INVALID, ERR_TYPE_WEBDLL, NULL,
iCurrent, iSyncId);
    (*Term.Delete)(pECB, iCurrent);
    return;
    }
    WriteZString(pECB, MakeMainMenuForm(iCurrent,
Term.pClientData[iCurrent].iSyncId) );
    return;
}
/* FUNCTION: void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function is the first command executed. It is executed
with the command
 *  CMD=Begin?Server=xxx from the http command line.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:SQL server must be specified, however the user and password
parameters are optional.
 *  The complete command line is CMD=Begin&Server=server&User=sa&Psw=&.
The & are used
 *  to separate parameters which is internet browser standard.
 */
void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    LPSTR pQueryString;

    pQueryString = pECB->lpszQueryString;
    if ( !GetKeyValue(pQueryString, "Server", szServer,
sizeof(szServer)) )
    {
    ErrorMessage(pECB, ERR_NO_SERVER_SPECIFIED, ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
    return;
    }
    if ( !GetKeyValue(pQueryString, "User", szUser, sizeof(szUser)) )
    strcpy(szUser, "sa");
    if ( !GetKeyValue(pQueryString, "Psw", szPassword,
sizeof(szPassword)) )
    strcpy(szPassword, "");
    if ( !GetKeyValue(pQueryString, "Db", szDatabase,
sizeof(szDatabase)) )
    strcpy(szDatabase, "tpcc");
    WriteZString(pECB, MakeWelcomeForm() );
    UNUSEDPARAM(iFormId);
    return;
}
/* FUNCTION: void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function process the passed in http command
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    switch( iFormId )
    {
    case WELCOME_FORM:
    return;
    case MAIN_MENU_FORM:
```

```
    return;
    case NEW_ORDER_FORM:
    ProcessNewOrderForm(pECB, iTermId, iSyncId);
    return;
    case PAYMENT_FORM:
    ProcessPaymentForm(pECB, iTermId, iSyncId);
    return;
    case DELIVERY_FORM:
    ProcessDeliveryForm(pECB, iTermId, iSyncId);
    return;
    case ORDER_STATUS_FORM:
    ProcessOrderStatusForm(pECB, iTermId, iSyncId);
    return;
    case STOCK_LEVEL_FORM:
    ProcessStockLevelForm(pECB, iTermId, iSyncId);
    return;
    }
}
/* FUNCTION: void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function frees all currently logged in terminal ids.
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:Use this function with caution, it may cause unpredictable
results
 *  if existing browsers attempt to use the web client with out
 *  beginning at the login screen for each client.
 */
void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    int i;
    EnterCriticalSection(&CriticalSection);
    for(i=0; i<Term.iAvailable; i++)
    {
    if ( Term.pClientData[i].inUse )
    (*Term.Delete)(pECB, i);
    }
    Term.iNext= 0;
    Term.iAvailable= 0;
    Term.iMasterSyncId= 1;
    if ( Term.pClientData )
    free(Term.pClientData);
    Term.pClientData= NULL;
    Term.bInit= FALSE;
    (*Term.Init)();
    if ( !(*Term.Allocate)() )
    {
    ErrorMessage(pECB, ERR_MAX_CONNECT_PARAM, ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
    return;
    }
    for(i=Term.iNext; i<Term.iAvailable; i++)
    Term.pClientData[i].inUse = 0;
    Term.pClientData[0].inUse = 1;
    LeaveCriticalSection(&CriticalSection);
    WriteZString(pECB, MakeWelcomeForm() );
    return;
}
/* FUNCTION: void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function causes an exit to the main menu
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
    WriteZString(pECB, MakeMainMenuForm(iTermId, iSyncId) );
    return;
}
/* FUNCTION: void NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK *pECB,
int iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE:This function returns to the browser the total number of
active terminal ids
 *
 * ARGUMENTS:intiFormIdunused
 *  intiTermIdid of calling browser, i.e. TERMID= from http command line
```

```
 *  intiSyncIdsync id of calling browser
 *  EXTENSION_CONTROL_BLOCK*pECBstructure pointer to passed in internet
 *  service information.
 * RETURNS:None
 *
 * COMMENTS:None
 */
void NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
{
    int i;
    intiTotal;
    // EnterCriticalSection(&CriticalSection);
    iTotal = 0;
    for(i=0; i<Term.iAvailable; i++)
    {
    if ( Term.pClientData[i].inUse )
    iTotal++;
    }
    // LeaveCriticalSection(&CriticalSection);
    h_printf(pECB, "Total Active Connections: %d", iTotal);
    return;
}
/* FUNCTION: void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char
*szStr)
 *
 * PURPOSE:This function is the low level output function. It writes a
string of text back to the
 *  client browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  char*szStrstring to display in the client browser.
 *
 * RETURNS:None
 *
 * COMMENTS:This function assumes that the string to written to the
client browser has
 *  been formatted in an HTML manner.
 */
void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr)
{
    FILE*fp;
    intlpbSize;
    intiSize;
    charszHeader[128];
    charszHeader1[128];
    lpbSize = strlen(szStr)+1;
    if ( bLog )
    {
    SYSTEMTIMEsystemTime;
    fp = fopen(szTpccLogPath, "ab");
    GetLocalTime(&systemTime);
    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",
    systemTime.wYear, systemTime.wMonth, systemTime.wDay,
    systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
    szStr);
    fclose(fp);
    }
    iSize = sprintf(szHeader, "200 Ok");
    sprintf(szHeader1, "Connection: keep-alive\r\nContent-type:
text/html\r\nContent-length: %d\r\n\r\n", lpbSize);
    (*pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_SEND_RESPONSE_HEADER, szHeader, &iSize, (LPDWORD)szHeader1);
    (*pECB->WriteClient)(pECB->ConnID, szStr, &lpbSize, 0);
    return;
}
/* FUNCTION: void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format,
...)
 *
 * PURPOSE:This function forms a high level printf for an HTML browser
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  char*formatprintf style format string
 *  ...other arguments as required by printf style format string.
 *
 * RETURNS:None
 *
 * COMMENTS:This function is mainly used for developmental support.
 */
static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...)
{
    char szBuff[512];
    char szTmp[512];
    va_list marker;
    va_start( marker, format );
    vsprintf(szTmp, format, marker);
    va_end( marker );
    wsprintf(szBuff, "<html>%s</html>", szTmp) + 1;
    WriteZString(pECB, szBuff);
    return;
}
/* FUNCTION: BOOL GetKeyValue(char *pQueryString, char *pKey, char
*pValue, int iMax)
```

```
 *
 * PURPOSE:This function parses a http formatted string for specific key
values.
 *
 * ARGUMENTS:char*pQueryStringhttp string from client browser
 *  char*pKeykey value to look for
 *  char*pValuecharacter array into which to place key's value
 *  intiMaxmaximum length of key value array.
 *
 * RETURNS:BOOLFALSEkey value not found
 *  TRUEkey valud found
 *
 *
 * COMMENTS:http keys are formatted either KEY=value& or KEY=value\0.
This DLL formats
 *  TPC-C input fields in such a manner that the keys can be extracted
in the
 *  above manner.
 */
static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue,
int iMax)
{
    char *ptr;
    if ( !(ptr=strstr(pQueryString, pKey)) )
    return FALSE;
    if ( !(ptr=strchr(ptr, '=')) )
    return FALSE;
    ptr++;
    iMax--;
    while( *ptr && *ptr != '&' && iMax)
    {
    *pValue++ = *ptr++;
    iMax--;
    }
    *pValue = 0;
    return TRUE;
}
/* FUNCTION: void TermInit(void)
 *
 * PURPOSE:This function initializes the client ternimal structure it is
called when the TPCC.DLL
 *  is first loaded by the inet service.
 *
 * ARGUMENTS:none
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void TermInit(void)
{
    if ( Term.bInit )
    return;
    Term.iNext= 0;
    Term.iMasterSyncId= 1;
    Term.iAvailable= 0;
    Term.pClientData= NULL;
    Term.bInit= TRUE;
    return;
}
/* FUNCTION: void TermRestore(void)
 *
 * PURPOSE:This function frees allocated resources associated with the
terminal structure.
 *
 * ARGUMENTS:none
 *
 * RETURNS:None
 *
 * COMMENTS:This function is called only with the inet service unloads
the TPCC.DLL
 *
 */
static void TermRestore(void)
{
    Term.iNext= 0;
    Term.iAvailable= 0;
    Term.iMasterSyncId= 0;
    if ( Term.pClientData )
    free(Term.pClientData);
    Term.pClientData= NULL;
    Term.bInit= FALSE;
    return;
}
/* FUNCTION: int TermAllocate(void)
 *
 * PURPOSE:This function allocates more terminal array entries in the
Term structure.
 *
 * ARGUMENTS:   None
 *
 * RETURNS:intTRUE or 1 if sucessfull
 *  intFALSE or 0 if terminal id cannot be allocated.
 *
```

```
 * COMMENTS:None
 *
 */
static int TermAllocate(void)
{
    Term.iAvailable += 32;
    if ( !Term.pClientData )
    Term.pClientData = (PCLIENTDATA)malloc(Term.iAvailable *
sizeof(CLIENTDATA));
    else
    Term.pClientData = (PCLIENTDATA)realloc(Term.pClientData,
Term.iAvailable * sizeof(CLIENTDATA));
    return ( Term.pClientData ) ? 1 : 0;
}
/* FUNCTION: int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char
*pQueryString)
 *
 * PURPOSE:This function assigns a terminal id which is used to identify
a client browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  char*pQueryStringhttp query string passed to this DLL.
 *
 * RETURNS:intassigned terminal id
 *  -1cannot assign id error occured.
 *
 *
 * COMMENTS:if the terminal id cannot be assigned it is because of
insufficient memory or the
 *  SQL connection cannot be allocated.
 *
 */
static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString)
{
    charszTmp[32];
    inti, iCurrent, iTotalConnections, iTickCount;
    EnterCriticalSection(&CriticalSection);
    for(i=0, iTotalConnections = 0; i<Term.iAvailable; i++)
    {
    if ( Term.pClientData[i].inUse )
    iTotalConnections++;
    }
    if ( iTotalConnections >= iMaxConnections )
    {
    for(iCurrent = 1, i=1, iTickCount = 0x7FFFFFFF; i<iMaxConnections;
i++)
    {
    if ( iTickCount > Term.pClientData[i].iTickCount )
    {
    iTickCount = Term.pClientData[i].iTickCount;
    iCurrent = i;
    }
    }
    }
    else
    {
    for(i=0; i<Term.iAvailable; i++)
    {
    if ( !Term.pClientData[i].inUse )
    break;
    }
    iCurrent = i;
    }
    if ( i == Term.iAvailable )
    {
    Term.iNext = Term.iAvailable;
    if ( !(*Term.Allocate)() )
    goto TermAddErr1;
    for(i=Term.iNext; i<Term.iAvailable; i++)
    Term.pClientData[i].inUse = 0;
    iCurrent = Term.iNext;
    }
    Term.pClientData[iCurrent].inUse = 1;
    if ( !GetKeyValue(pQueryString, "w_id", szTmp, sizeof(szTmp)) )
    goto TermAddErr1;
    Term.pClientData[iCurrent].w_id = (short)atoi(szTmp);
    if ( !GetKeyValue(pQueryString, "d_id", szTmp, sizeof(szTmp)) )
    goto TermAddErr1;
    Term.pClientData[iCurrent].d_id = atoi(szTmp);
    Term.pClientData[iCurrent].iTickCount = GetTickCount();
    Term.pClientData[iCurrent].iSyncId = Term.iMasterSyncId++;
    if ( Init(pECB, iCurrent, Term.pClientData[iCurrent].iSyncId,
szServer, szUser, szPassword, szDatabase) )
    {
    (*Term.Delete)(pECB, iCurrent);
    goto TermAddErr1;
    }
    LeaveCriticalSection(&CriticalSection);
    return iCurrent;
TermAddErr1:
    LeaveCriticalSection(&CriticalSection);
    return -1;//terminal unsuccessfully added
}
/* FUNCTION: void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id)
```

```
 *
 * PURPOSE:This function makes a terminal entry in the Term array
available for reuse.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intidTerminal id of client exiting
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id)
{
    if ( id >= 0 && id < Term.iAvailable )
    {
    Close(pECB, id, -1);
    Term.pClientData[id].inUse = 0;
    }
    return;
}
/* FUNCTION: BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, char *szServer, char *szUser, char *szPassword, char
*szDatabase)
 *
 * PURPOSE:This function initializes the sql connection for use.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdid of browser client that this connection is for.
 *  intiSyncIdsync id for this client session
 *  char*szServersql server name
 *  char*szUseruser name
 *  char*szPassworduser password
 *  char*szDatabasedatabase to use
 *
 * RETURNS:BOOLFALSEif successfull
 *  TRUEif an error occurs and connection cannot be established.
 *
 * COMMENTS:None
 *
 */
BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, char
*szServer, char *szUser, char *szPassword, char *szDatabase)
{
    charszApp[32];
    charserver[256];
    chardatabase[256];
    charuser[256];
    charpassword[256];
    sprintf(szApp, "TPCC:%ld", (int)iTermId);
    Term.pClientData[iTermId].dbproc = NULL;
    sprintf(szApp, "TPCC:%ld", (int)iTermId);
    Term.pClientData[iTermId].dbproc = NULL;
    strcpy(server, szServer);
    strcpy(database, szDatabase);
    strcpy(user, szUser);
    strcpy(password, szPassword);
    if ( SQLOpenConnection(pECB, iTermId, iSyncId,
&Term.pClientData[iTermId].dbproc, server, database, user, password,
szApp, &Term.pClientData[iTermId].spid) )
    {
    ErrorMessage(pECB, ERR_SQL_OPEN_CONNECTION, ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
    return TRUE;
    }
    return FALSE;
}
/* FUNCTION: BOOL Close(EXTENSION_CONTROL_BLOCK*pECB, int iTermId, int
iSyncId)
 *
 * PURPOSE:This function closes the sql connection for use.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdid of browser client that this connection is for.
 *  intiSyncIdsync id of client browser
 *
 * RETURNS:BOOLFALSEif successfull
 *  TRUEif an error occurs and connection cannot be terminated.
 *
 * COMMENTS:None
 *
 */
static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId)
{
    PECBINFOpEcbInfo;
    if (Term.pClientData[iTermId].dbproc != NULL)
    {
    if ( (pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)))
    {
    pEcbInfo->iTermId = -1;
    pEcbInfo->iSyncId = -1;
    free(pEcbInfo);//free up user info
```

```
    }
    return SQLCloseConnection(pECB, Term.pClientData[iTermId].dbproc);
    }
    UNUSEDPARAM(iSyncId);
}
/* FUNCTION: void FormatString(char *szDest, char *szPic, char *szSrc)
 *
 * PURPOSE:This function formats a character string for inclusion in the
 *  HTML formatted page being constructed.
 *
 * ARGUMENTS:char*szDestDestination buffer where formatted string is to
be placed
 *  char*szPicpicture string which describes how character value is to be
 *  formatted.
 *  char*szSrccharacter string value.
 *
 * RETURNS:None
 *
 * COMMENTS:This functions is used to format TPC-C phone and zip value
strings.
 *
 */
static void FormatString(char *szDest, char *szPic, char *szSrc)
{
    while( *szPic )
    {
    if ( *szPic == 'X' )
    {
    if ( *szSrc )
    *szDest++ = *szSrc++;
    else
    *szDest++ = ' ';
    }
    else
    *szDest++ = *szPic;
    szPic++;
    }
    *szDest = 0;
    return;
}
/* FUNCTION: char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL
bInput)
 *
 * PURPOSE:This function constructs the Stock Level HTML page.
 *
 * ARGUMENTS:intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
 *  BOOLbInputTRUE if form is being constructed for input else FALSE
 *
 * RETURNS:char *A pointer to buffer inside client structure where HTML
form is built.
 *
 * COMMENTS:The internal client buffer is created when the terminal id
is assigned and should not
 *  be freed except when the client terminal id is no longer needed.
 */
static char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput)
{
    char*szForm;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;
    Term.pClientData[iTermId].stockLevelData.w_id=
(short)Term.pClientData[iTermId].w_id;
    Term.pClientData[iTermId].stockLevelData.d_id=
(short)Term.pClientData[iTermId].d_id;
    Term.pClientData[iTermId].stockLevelData.num_deadlocks= 0;
    strcpy(szForm, "<HTML><HEAD><TITLE>TPC-C Stock
Level</TITLE></HEAD>");
    strcat(szForm, "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    if ( bInput )
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", STOCK_LEVEL_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    strcat(szForm, "<PRE>                                Stock-
Level<BR>");
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d   District:
%2.2d<BR><BR>", Term.pClientData[iTermId].stockLevelData.w_id,
Term.pClientData[iTermId].stockLevelData.d_id);
    if ( bInput )
    {
    strcat(szForm,"Stock Level Threshold: <INPUT NAME=\"TT*\"
SIZE=2><BR><BR>"
    "low stock:    <BR><HR>"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">" );
    }
    else
    {
    wsprintf(szForm+strlen(szForm), "Stock Level Threshold:
%2.2d<BR><BR>", Term.pClientData[iTermId].stockLevelData.thresh_hold);
```

```
    wsprintf(szForm+strlen(szForm), "low stock: %3.3d</PRE><BR><HR>",
Term.pClientData[iTermId].stockLevelData.low_stock);
    strcat(szForm,"<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">" );
    }
    strcat(szForm, "</FORM></HTML>");
    return szForm;
}
/* FUNCTION: char *MakeMainMenuForm(int iTermId, int iSyncId)
 *
 * PURPOSE:This function
 *
 * ARGUMENTS:intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
 *
 * RETURNS:char *A pointer to buffer inside client structure where HTML
form is built.
 *
 * COMMENTS:The internal client buffer is created when the terminal id
is assigned and should not
 *  be freed except when the client terminal id is no longer needed.
 */
static char *MakeMainMenuForm(int iTermId, int iSyncId)
{
    char*szForm;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;
    strcpy(szForm,"<HTML><HEAD><TITLE>TPC-C Main
Menu</TITLE></HEAD><BODY>"
    "Select Desired Transaction.<BR><HR>"
    "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", MAIN_MENU_FORM);
    strcat(szForm, "<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">"
    "</FORM>"
    "</HTML>" );
    return szForm;
}
/* FUNCTION: char *MakeWelcomeForm(void)
 *
 * PURPOSE:This function
 *
 * ARGUMENTS:   None
 *
 * RETURNS:char *A pointer to the static HTML welcome form.
 *
 * COMMENTS:The welcome form is static.
 */
static char *MakeWelcomeForm(void)
{
    return szWelcomeForm;
}
/* FUNCTION: char *MakeNewOrderForm(int iTermId, BOOL bInput, BOOL
bValid)
 *
 * PURPOSE:This function
 *
 * ARGUMENTS:intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
 *  BOOLbInputTRUE if form is being constructed for input else FALSE
 *  BOOLbValidTRUE if NeworderData valid, ELSE FALSE effects output only
 *
 * RETURNS:char *A pointer to buffer inside client structure where HTML
form is built.
 *
 * COMMENTS:The internal client buffer is created when the terminal id
is assigned and should not
 *  be freed except when the client terminal id is no longer needed.
 */
static char *MakeNewOrderForm(int iTermId, int iSyncId, BOOL bInput,
BOOL bValid)
{
    char*szForm;
    charszName[146];
    charszCredit[14];
    inti;
    szForm = (char *)Term.pClientData[iTermId].szBuffer;
    Term.pClientData[iTermId].newOrderData.w_id =
Term.pClientData[iTermId].w_id;
    strcpy(szForm,"<HTML>"
```

```
    "<HEAD><TITLE>TPC-C New Order</TITLE></HEAD><BODY>"
    "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">" );
    if ( bInput )
    {
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    }
    else
    {
    if ( bValid )
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    else
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">", ERR_BAD_ITEM_ID);
    }
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", NEW_ORDER_FORM);
    wsprintf(szForm+strlen(szForm),"<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    strcat(szForm, "<PRE>                                New Order<BR>");
    if ( bInput )
    {
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d   District: <INPUT
NAME=\"DID*\" SIZE=1>                        Date:<BR>",
Term.pClientData[iTermId].newOrderData.w_id);
    strcat(szForm,"Customer:  <INPUT NAME=\"CID*\" SIZE=4>
Name:                    Credit:      %Disc:<BR>"
    "Order Number:           Number of Lines:           W_tax:
D_tax:<BR><BR>"
    " Supp_W  Item_Id  Item Name                 Qty  Stock  B/G
Price    Amount<BR>"
    " <INPUT NAME=\"SP00*\" SIZE=4>  <INPUT NAME=\"IID00*\"
SIZE=6>                           <INPUT NAME=\"Qty00*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP01*\" SIZE=4>  <INPUT NAME=\"IID01*\"
SIZE=6>                           <INPUT NAME=\"Qty01*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP02*\" SIZE=4>  <INPUT NAME=\"IID02*\"
SIZE=6>                           <INPUT NAME=\"Qty02*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP03*\" SIZE=4>  <INPUT NAME=\"IID03*\"
SIZE=6>                           <INPUT NAME=\"Qty03*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP04*\" SIZE=4>  <INPUT NAME=\"IID04*\"
SIZE=6>                           <INPUT NAME=\"Qty04*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP05*\" SIZE=4>  <INPUT NAME=\"IID05*\"
SIZE=6>                           <INPUT NAME=\"Qty05*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP06*\" SIZE=4>  <INPUT NAME=\"IID06*\"
SIZE=6>                           <INPUT NAME=\"Qty06*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP07*\" SIZE=4>  <INPUT NAME=\"IID07*\"
SIZE=6>                           <INPUT NAME=\"Qty07*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP08*\" SIZE=4>  <INPUT NAME=\"IID08*\"
SIZE=6>                           <INPUT NAME=\"Qty08*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP09*\" SIZE=4>  <INPUT NAME=\"IID09*\"
SIZE=6>                           <INPUT NAME=\"Qty09*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP10*\" SIZE=4>  <INPUT NAME=\"IID10*\"
SIZE=6>                           <INPUT NAME=\"Qty10*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP11*\" SIZE=4>  <INPUT NAME=\"IID11*\"
SIZE=6>                           <INPUT NAME=\"Qty11*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP12*\" SIZE=4>  <INPUT NAME=\"IID12*\"
SIZE=6>                           <INPUT NAME=\"Qty12*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP13*\" SIZE=4>  <INPUT NAME=\"IID13*\"
SIZE=6>                           <INPUT NAME=\"Qty13*\" SIZE=1><BR>"
    " <INPUT NAME=\"SP14*\" SIZE=4>  <INPUT NAME=\"IID14*\"
SIZE=6>                           <INPUT NAME=\"Qty14*\" SIZE=1><BR>"
    "Execution Status:
Total:<BR><HR>"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">"
    "</FORM>"
    "</HTML>" );
    }
    else
    {
    if ( bValid )
    {
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d   District:
%2.2d                       Date: %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d
<BR>",
    Term.pClientData[iTermId].newOrderData.w_id,
    Term.pClientData[iTermId].newOrderData.d_id,
    Term.pClientData[iTermId].newOrderData.o_entry_d.day,
    Term.pClientData[iTermId].newOrderData.o_entry_d.month,
    Term.pClientData[iTermId].newOrderData.o_entry_d.year,
    Term.pClientData[iTermId].newOrderData.o_entry_d.hour,
    Term.pClientData[iTermId].newOrderData.o_entry_d.minute,
    Term.pClientData[iTermId].newOrderData.o_entry_d.second);
    }
    else
    {
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d   District:
%2.2d                       Date:<BR>",
    Term.pClientData[iTermId].newOrderData.w_id,
    Term.pClientData[iTermId].newOrderData.d_id);
    }
    FormatHTMLString(szName,
Term.pClientData[iTermId].newOrderData.c_last, 16),
```

```
    FormatHTMLString(szCredit,
Term.pClientData[iTermId].newOrderData.c_credit, 2);
    wsprintf(szForm+strlen(szForm), "Customer:  %4.4d   Name: %s
Credit: %s   ",
    Term.pClientData[iTermId].newOrderData.c_id, szName, szCredit);
    if ( bValid )
    {
    sprintf(szForm+strlen(szForm), "%%Disc: %5.2f            <BR>",
Term.pClientData[iTermId].newOrderData.c_discount);
    sprintf(szForm+strlen(szForm), "Order Number: %8.8d  Number of
Lines: %2.2d        W_tax: %5.2f   D_tax: %5.2f  <BR><BR>",
    Term.pClientData[iTermId].newOrderData.o_id,
    Term.pClientData[iTermId].newOrderData.o_ol_cnt,
    Term.pClientData[iTermId].newOrderData.w_tax,
    Term.pClientData[iTermId].newOrderData.d_tax);
    strcat(szForm, " Supp_W  Item_Id  Item Name                 Qty
Stock  B/G  Price    Amount<BR>");
    for(i=0; i<Term.pClientData[iTermId].newOrderData.o_ol_cnt; i++)
    {
    FormatHTMLString(szName,
Term.pClientData[iTermId].newOrderData.Ol[i].ol_i_name, 24);
    sprintf(szForm+strlen(szForm), "  %4.4d   %6.6d   %s  %2.2d
%3.3d    %1.1s   $%6.2f  $%7.2f  <BR>",
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_supply_w_id,
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_i_id,
    szName,
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_quantity,
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_stock,
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_brand_generic,
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_i_price,
    Term.pClientData[iTermId].newOrderData.Ol[i].ol_amount );
    }
    }
    else
    {
    strcat(szForm, "%Disc:<BR>");
    sprintf(szForm+strlen(szForm), "Order Number: %8.8d  Number of
Lines:              W_tax:         D_tax:<BR><BR>",
    Term.pClientData[iTermId].newOrderData.o_id);
    strcat(szForm, " Supp_W  Item_Id  Item Name                 Qty
Stock  B/G  Price    Amount<BR>");
    i = 0;
    }
    for(; i<15; i++)
    strcat(szForm, "<BR>");
    if ( bValid )
    {
    sprintf(szForm+strlen(szForm), "Execution Status:
%24.24s                  Total:  $%8.2f  ",
    Term.pClientData[iTermId].newOrderData.execution_status,
    Term.pClientData[iTermId].newOrderData.total_amount);
    }
    else
    {
    sprintf(szForm+strlen(szForm), "Execution Status:
%24.24s                  Total:",
    Term.pClientData[iTermId].newOrderData.execution_status);
    }
    strcat(szForm,"</PRE><HR><BR>"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">" );
    strcat(szForm, "</FORM></HTML>");
    }
    return szForm;
}
/* FUNCTION: char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput)
 *
 * PURPOSE:This function
 *
 * ARGUMENTS:intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
 *  BOOLbInputTRUE if form is being constructed for input else FALSE
 *
 * RETURNS:char *A pointer to buffer inside client structure where HTML
form is built.
 *
 * COMMENTS:The internal client buffer is created when the terminal id
is assigned and should not
 *  be freed except when the client terminal id is no longer needed.
 */
static char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput)
{
    char*szForm;
    char*ptr;
    charszTmp[64];
    charszW_Zip[26];
    charszD_Zip[26];
    charszC_Zip[26];
    charszC_Phone[26];
    charszTmpStr1[122];
    charszTmpStr2[122];
    charszTmpStr3[122];
```

```
    charszTmpStr4[122];
    inti;
    intl;
    char*szZipPic = "XXXXX-XXXX";
    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    Term.pClientData[iTermId].paymentData.w_id =
Term.pClientData[iTermId].w_id;
    strcpy(szForm,"<HTML><HEAD><TITLE>TPC-C Payment</TITLE></HEAD><BODY>"
    "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    if ( bInput )
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", PAYMENT_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    strcat(szForm,"<PRE>                                    Payment<BR>");
    if ( bInput )
    strcat(szForm,"Date:<BR><BR>" );
    else
    {
    wsprintf(szForm+strlen(szForm), "Date: %2.2d-%2.2d-%4.4d
%2.2d:%2.2d:%2.2d <BR><BR>",
    Term.pClientData[iTermId].paymentData.h_date.day,
    Term.pClientData[iTermId].paymentData.h_date.month,
    Term.pClientData[iTermId].paymentData.h_date.year,
    Term.pClientData[iTermId].paymentData.h_date.hour,
    Term.pClientData[iTermId].paymentData.h_date.minute,
    Term.pClientData[iTermId].paymentData.h_date.second);
    }
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d",
Term.pClientData[iTermId].paymentData.w_id);
    if ( bInput )
    {
    strcat(szForm,"                          District: <INPUT
NAME=\"DID*\" SIZE=1><BR><BR><BR><BR><BR>"
    "Customer: <INPUT NAME=\"CID*\" SIZE=4>"
    "Cust-Warehouse: <INPUT NAME=\"CWI*\" SIZE=4>  "
    "Cust-District: <INPUT NAME=\"CDI*\" SIZE=1><BR>"
    "Name:                    <INPUT NAME=\"CLT*\"
SIZE=16>                  Since:<BR>"
    "                                          Credit:<BR>"
    "                                          Disc:<BR>"
    "                                          Phone:<BR><BR>"
    "Amount Paid:          $<INPUT NAME=\"HAM*\" SIZE=7>      New Cust
Balance:<BR>"
    "Credit Limit:<BR><BR>Cust-Data: <BR><BR><BR><BR></PRE><HR>"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\"><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">"
    "</BODY></FORM></HTML>" );
    }
    else
    {
    sprintf(szForm+strlen(szForm),"                              District:
%2.2d<BR>",
    Term.pClientData[iTermId].paymentData.d_id);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.w_street_1, 20);
    FormatHTMLString(szTmpStr2,
Term.pClientData[iTermId].paymentData.d_street_1, 20);
    sprintf(szForm+strlen(szForm),"%s                        %s<BR>",
szTmpStr1, szTmpStr2);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.w_street_2, 20);
    FormatHTMLString(szTmpStr2,
Term.pClientData[iTermId].paymentData.d_street_2, 20);
    sprintf(szForm+strlen(szForm),"%s                        %s<BR>",
szTmpStr1, szTmpStr2);
    FormatString(szW_Zip, szZipPic,
Term.pClientData[iTermId].paymentData.w_zip);
    FormatString(szD_Zip, szZipPic,
Term.pClientData[iTermId].paymentData.d_zip);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.w_city, 20);
    FormatHTMLString(szTmpStr2,
Term.pClientData[iTermId].paymentData.w_state, 2);
    FormatHTMLString(szTmpStr3,
Term.pClientData[iTermId].paymentData.d_city, 20);
    FormatHTMLString(szTmpStr4,
Term.pClientData[iTermId].paymentData.d_state, 2);
    wsprintf(szForm+strlen(szForm), "%s %s %10.10s       %s %s
%10.10s<BR><BR>",
    szTmpStr1, szTmpStr2, szW_Zip, szTmpStr3, szTmpStr4, szD_Zip );
    wsprintf(szForm+strlen(szForm), "Customer: %4.4d  Cust-Warehouse:
%4.4d  Cust-District: %2.2d<BR>",
    Term.pClientData[iTermId].paymentData.c_id,
    Term.pClientData[iTermId].paymentData.c_w_id,
    Term.pClientData[iTermId].paymentData.c_d_id);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.c_first, 16);
    FormatHTMLString(szTmpStr2,
Term.pClientData[iTermId].paymentData.c_middle, 2);
    FormatHTMLString(szTmpStr3,
Term.pClientData[iTermId].paymentData.c_last, 16);
```

```
    wsprintf(szForm+strlen(szForm), "Name:   %s %s %s     Since:  %2.2d-
%2.2d-%4.4d<BR>",
    szTmpStr1, szTmpStr2, szTmpStr3,
    Term.pClientData[iTermId].paymentData.c_since.day,
    Term.pClientData[iTermId].paymentData.c_since.month,
    Term.pClientData[iTermId].paymentData.c_since.year);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.c_street_1, 20);
    FormatHTMLString(szTmpStr2,
Term.pClientData[iTermId].paymentData.c_credit, 2);
        wsprintf(szForm+strlen(szForm), "        %s
Credit: %s<BR>", szTmpStr1, szTmpStr2);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.d_street_2, 20);
        sprintf(szForm+strlen(szForm), "        %s
%%Disc:  %5.2f<BR>",
    szTmpStr1, Term.pClientData[iTermId].paymentData.c_discount);
    FormatString(szC_Zip, szZipPic,
Term.pClientData[iTermId].paymentData.c_zip);
    FormatString(szC_Phone, "XXXXXX-XXX-XXX-XXXX",
Term.pClientData[iTermId].paymentData.c_phone);
    FormatHTMLString(szTmpStr1,
Term.pClientData[iTermId].paymentData.c_city, 20);
    FormatHTMLString(szTmpStr2,
Term.pClientData[iTermId].paymentData.c_state, 2);
    wsprintf(szForm+strlen(szForm), "        %s %s %10.10s       Phone:
%-19.19s<BR><BR>",
    szTmpStr1, szTmpStr2, szC_Zip, szC_Phone );
    sprintf(szForm+strlen(szForm), "Amount Paid:          $%7.2f
New Cust Balance: $%14.2f<BR>",
    Term.pClientData[iTermId].paymentData.h_amount,
    Term.pClientData[iTermId].paymentData.c_balance);
    sprintf(szForm+strlen(szForm), "Credit Limit:   $%13.2f<BR><BR>",
    Term.pClientData[iTermId].paymentData.c_credit_lim);
    ptr = Term.pClientData[iTermId].paymentData.c_credit;
    if ( *ptr == 'B' && *(ptr+1) == 'C' )
    {
    ptr = Term.pClientData[iTermId].paymentData.c_data;
    l = strlen( ptr ) / 50;
    for(i=0; i<4; i++, ptr += 50)
    {
    if ( i <= l )
    UtilStrCpy(szTmp, ptr, 50);
    else
    szTmp[0] = 0;
    if ( !i )
    {
    FormatHTMLString(szTmpStr1, szTmp, 50);
    wsprintf(szForm+strlen(szForm), "Cust-Data: %s<BR>", szTmpStr1);
    }
    else
    {
    FormatHTMLString(szTmpStr1, szTmp, 50);
    wsprintf(szForm+strlen(szForm), "           %s<BR>", szTmpStr1);
    }
    }
    }
    else
    strcat(szForm, "Cust-Data: <BR><BR><BR><BR>");
    strcat(szForm,"</PRE><HR><BR>"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">"
    "</BODY></FORM></HTML>");
    }
    return szForm;
}
/* FUNCTION: char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL
bInput)
 *
 * PURPOSE:This function
 *
 * ARGUMENTS:intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
 *  BOOLbInputTRUE if form is being constructed for input else FALSE
 *
 * RETURNS:char *A pointer to buffer inside client structure where HTML
form is built.
 *
 * COMMENTS:The internal client buffer is created when the terminal id
is assigned and should not
 *  be freed except when the client terminal id is no longer needed.
 */
static char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput)
{
    char*szForm;
    charc_first[98];
    charc_middle[14];
    charc_last[98];
    inti;
    szForm = (char *)Term.pClientData[iTermId].szBuffer;
    Term.pClientData[iTermId].orderStatusData.w_id =
Term.pClientData[iTermId].w_id;
```

```
    strcpy(szForm,"<HTML><HEAD><TITLE>TPC-C Order-
Status</TITLE></HEAD><BODY>"
    "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    if ( bInput )
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", ORDER_STATUS_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    strcat(szForm,"<PRE>                                Order-Status<BR>"
);
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d   ",
Term.pClientData[iTermId].orderStatusData.w_id);
    if ( bInput )
    {
    strcat(szForm,"District: <INPUT NAME=\"DID*\" SIZE=1><BR>"
    "Customer: <INPUT NAME=\"CID*\" SIZE=4>   Name:
<INPUT NAME=\"CLT*\" SIZE=23><BR>"
    "Cust-Balance:<BR><BR>"
    "Order-Number:            Entry-Date:                       Carrier-
Number:<BR>"
    "Supply-W     Item-Id    Qty     Amount      Delivery-Date<BR></PRE>"
    "<HR><INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\"><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">"
    "</BODY></FORM></HTML>" );
    }
    else
    {
    wsprintf(szForm+strlen(szForm), "District: %2.2d<BR>",
Term.pClientData[iTermId].orderStatusData.d_id);
    FormatHTMLString(c_first,
Term.pClientData[iTermId].orderStatusData.c_first, 16);
    FormatHTMLString(c_middle,
Term.pClientData[iTermId].orderStatusData.c_middle, 2);
    FormatHTMLString(c_last,
Term.pClientData[iTermId].orderStatusData.c_last, 16);
    wsprintf(szForm+strlen(szForm), "Customer: %4.4d   Name: %s %s
%s<BR>",
    Term.pClientData[iTermId].orderStatusData.c_id, c_first, c_middle,
c_last);
    sprintf(szForm+strlen(szForm), "Cust-Balance: $%9.2f<BR><BR>",
    Term.pClientData[iTermId].orderStatusData.c_balance);
    wsprintf(szForm+strlen(szForm), "Order-Number: %8.8d   Entry-Date:
%2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d   Carrier-Number: %2.2d<BR>",
    Term.pClientData[iTermId].orderStatusData.o_id,
    Term.pClientData[iTermId].orderStatusData.o_entry_d.day,
    Term.pClientData[iTermId].orderStatusData.o_entry_d.month,
    Term.pClientData[iTermId].orderStatusData.o_entry_d.year,
    Term.pClientData[iTermId].orderStatusData.o_entry_d.hour,
    Term.pClientData[iTermId].orderStatusData.o_entry_d.minute,
    Term.pClientData[iTermId].orderStatusData.o_entry_d.second,
    Term.pClientData[iTermId].orderStatusData.o_carrier_id);
    strcat(szForm+strlen(szForm), "Supply-W     Item-Id    Qty
Amount      Delivery-Date<BR>");
    for(i=0; i<Term.pClientData[iTermId].orderStatusData.o_ol_cnt; i++)
    {
    sprintf(szForm+strlen(szForm), "  %4.4d       %6.6d     %2.2d
$%8.2f      %2.2d-%2.2d-%4.4d<BR>",

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_supply_
w_id,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_i_id,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_quantit
y,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_amount,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliver
y_d.day,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliver
y_d.month,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliver
y_d.year);
    }
    strcat(szForm,"<BR></PRE><HR><INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">"
    "</BODY></FORM></HTML>" );
    }
    return szForm;
}
/* FUNCTION: char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL
bInput, BOOL bSuccess)
 *
 * PURPOSE:This function
 *
 * ARGUMENTS:intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
```

```
 *  BOOLbInputTRUE if form is being constructed for input else FALSE
 *  BOOLbSuccess TRUE if Delivery succeeded else FALSE
 *
 * RETURNS:char *A pointer to buffer inside client structure where HTML
form is built.
 *
 * COMMENTS:The internal client buffer is created when the terminal id
is assigned and should not
 *  be freed except when the client terminal id is no longer needed.
 */
static char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput,
BOOL bSuccess)
{
    char*szForm;
    szForm = (char *)Term.pClientData[iTermId].szBuffer;
    Term.pClientData[iTermId].deliveryData.w_id =
Term.pClientData[iTermId].w_id;
    strcpy( szForm,"<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD><BODY>"
    "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    if ( bInput )
    {
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI*\" VALUE=\"\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    }
    else
    {
    if ( !bSuccess )
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">", ERR_TYPE_DELIVERY_POST);
    else
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
    }
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", DELIVERY_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    strcat(szForm,"<PRE>                                    Delivery<BR>" );
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d<BR><BR>",
Term.pClientData[iTermId].deliveryData.w_id);
    if ( bInput )
    strcat( szForm, "Carrier Number: <INPUT NAME=\"OCD*\"
SIZE=1><BR><BR>");
    else
    {
    wsprintf(szForm+strlen(szForm), "Carrier Number: %2.2d<BR><BR>",
    Term.pClientData[iTermId].deliveryData.o_carrier_id);
    }
    if ( bInput )
    {
    strcat( szForm, "Execution Status:<BR></PRE>"
    "<HR><INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">" );
    }
    else
    {
    wsprintf(szForm+strlen(szForm), "Execution Status:
%25.25s<BR></PRE>",
    Term.pClientData[iTermId].deliveryData.execution_status);
    strcat(szForm,"<HR><INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..NewOrder..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Payment..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Delivery..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-Status..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
    "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">" );
    }
    strcat( szForm,"</BODY></FORM></HTML>" );
    return szForm;
}
/* FUNCTION: void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK*pECB, int
iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from the new
order form
 *  filling in the required input variables. it then calls the
SQLNewOrder
 *  transaction, constructs the output form and writes it back to client
 *  browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdclient browser terminal id
 *  intiSyncId client browser sync id
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK*pECB, int
iTermId, int iSyncId)
{
```

```
    intiRc;
    intiError;
    PECBINFOpEcbInfo;
    memset(&Term.pClientData[iTermId].newOrderData, 0,
sizeof(NEW_ORDER_DATA));
    Term.pClientData[iTermId].newOrderData.w_id =
Term.pClientData[iTermId].w_id;
    if ( (iError=GetNewOrderData(pECB->lpszQueryString,
&Term.pClientData[iTermId].newOrderData)) != ERR_SUCCESS )
    {
    ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    return;
    }
    iRc = SQLNewOrder(pECB, iTermId, iSyncId,
Term.pClientData[iTermId].dbproc,
&Term.pClientData[iTermId].newOrderData, iDeadlockRetry);
#ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if ( bConnectionPooling && iRc != -3 )
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
#endif
    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
        pEcbInfo->bFailed)
    return;
    if ( iRc < 0 )
    ErrorMessage(pECB, ERR_NEW_ORDER_NOT_PROCESSED, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    else
    WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, FALSE,
(BOOL)iRc) );
    return;
}
/* FUNCTION: void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from the
payment form
 *  filling in the required input variables. It then calls the SQLPayment
 *  transaction, constructs the output form and writes it back to client
 *  browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdclient browser terminal id
 *  intiSyncId client browser sync id
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    intiRc;
    intiError;
    PECBINFOpEcbInfo;
    memset(&Term.pClientData[iTermId].paymentData, 0,
sizeof(PAYMENT_DATA));
    Term.pClientData[iTermId].paymentData.w_id =
Term.pClientData[iTermId].w_id;
    if ( (iError=GetPaymentData(pECB->lpszQueryString,
&Term.pClientData[iTermId].paymentData)) != ERR_SUCCESS )
    {
    ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    return;
    }
    iRc = SQLPayment(pECB, iTermId, iSyncId,
Term.pClientData[iTermId].dbproc,
&Term.pClientData[iTermId].paymentData, iDeadlockRetry);
#ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if ( bConnectionPooling && iRc != -3 )
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
#endif
    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
        pEcbInfo->bFailed)
    return;
    if ( iRc == 0 )
    ErrorMessage(pECB, ERR_PAYMENT_INVALID_CUSTOMER, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    else if ( iRc < 0 )
    ErrorMessage(pECB, ERR_PAYMENT_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
    else
    WriteZString(pECB, MakePaymentForm(iTermId, iSyncId, FALSE) );
    return;
}
/* FUNCTION: void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from the
Order Status
 *  form filling in the required input variables. It then calls the
 *  SQLOrderStatus transaction, constructs the output form and writes it
 *  back to client browser.
```

```
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdclient browser terminal id
 *  intiSyncId client browser sync id
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    intiRc;
    intiError;
    PECBINFOpEcbInfo;
    memset(&Term.pClientData[iTermId].orderStatusData, 0,
sizeof(ORDER_STATUS_DATA));
    Term.pClientData[iTermId].orderStatusData.w_id =
Term.pClientData[iTermId].w_id;
    if ( (iError=GetOrderStatusData(pECB->lpszQueryString,
&Term.pClientData[iTermId].orderStatusData)) != ERR_SUCCESS )
    {
    ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    return;
    }
    iRc = SQLOrderStatus(pECB, iTermId, iSyncId,
Term.pClientData[iTermId].dbproc,
&Term.pClientData[iTermId].orderStatusData, iDeadlockRetry);
#ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if ( bConnectionPooling && iRc != -3 )
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
#endif
    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
        pEcbInfo->bFailed)
    return;
    if ( iRc == 0 )
    ErrorMessage(pECB, ERR_NOSUCH_CUSTOMER, ERR_TYPE_WEBDLL, NULL,
iTermId, iSyncId);
    else if ( iRc < 0 )
    ErrorMessage(pECB, ERR_ORDER_STATUS_NOT_PROCESSED, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    else
    WriteZString(pECB, MakeOrderStatusForm(iTermId, iSyncId, FALSE) );
    return;
}
/* FUNCTION: void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from the
delivery form
 *  filling in the required input variables. It then calls the
PostDeliveryInfo
 *  Api, The client is then informed that the transaction has been
posted.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdclient browser terminal id
 *  intiSyncIdclinet browser sync id
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    charszTmp[26];
    BOOLbSuccess;
    memset(&Term.pClientData[iTermId].deliveryData, 0,
sizeof(DELIVERY_DATA));
    Term.pClientData[iTermId].deliveryData.w_id =
Term.pClientData[iTermId].w_id;
    if ( !GetKeyValue(pECB->lpszQueryString, "OCD*", szTmp,
sizeof(szTmp)) )
    {
    ErrorMessage(pECB, ERR_DELIVERY_MISSING_OCD_KEY, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    return;
    }
    if ( !IsNumeric(szTmp) )
    {
    ErrorMessage(pECB, ERR_DELIVERY_CARRIER_INVALID, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    return;
    }
    Term.pClientData[iTermId].deliveryData.o_carrier_id= atoi(szTmp);
    if ( Term.pClientData[iTermId].deliveryData.o_carrier_id > 10 ||
Term.pClientData[iTermId].deliveryData.o_carrier_id < 1 )
    {
    ErrorMessage(pECB, ERR_DELIVERY_CARRIER_ID_RANGE, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
```

```
    return;
    }
    //post delivery info
    if ( PostDeliveryInfo(Term.pClientData[iTermId].deliveryData.w_id,
Term.pClientData[iTermId].deliveryData.o_carrier_id) )
    {
    strcpy(Term.pClientData[iTermId].deliveryData.execution_status,
"Delivery Post Failed");
    bSuccess = FALSE;
    }
    else
    {
    strcpy(Term.pClientData[iTermId].deliveryData.execution_status,
"Delivery has been queued.");
    bSuccess = TRUE;
    }
    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, FALSE,
bSuccess) );
    return;
}
/* FUNCTION: void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB,
int iTermId, int iSyncId)
 *
 * PURPOSE:This function gets and validates the input data from the
Stock Level
 *  form filling in the required input variables. It then calls the
 *  SQLStockLevel transaction, constructs the output form and writes it
 *  back to client browser.
 *
 * ARGUMENTS:EXTENSION_CONTROL_BLOCK*pECBpassed in structure pointer
from inetsrv.
 *  intiTermIdclient browser terminal id
 *  intiSyncIdclient browser sync id
 *
 * RETURNS:None
 *
 * COMMENTS:None
 *
 */
static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    charszTmp[26];
    intiRc;
    PECBINFOpEcbInfo;
    memset(&Term.pClientData[iTermId].stockLevelData, 0,
sizeof(STOCK_LEVEL_DATA));
    Term.pClientData[iTermId].stockLevelData.w_id =
Term.pClientData[iTermId].w_id;
    Term.pClientData[iTermId].stockLevelData.d_id =
Term.pClientData[iTermId].d_id;
    if ( !GetKeyValue(pECB->lpszQueryString, "TT*", szTmp,
sizeof(szTmp)) )
    {
    ErrorMessage(pECB, ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    return;
    }
    if ( !IsNumeric(szTmp) )
    {
    ErrorMessage(pECB, ERR_STOCKLEVEL_THRESHOLD_INVALID,
ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    return;
    }
    Term.pClientData[iTermId].stockLevelData.thresh_hold = atoi(szTmp);
    if ( Term.pClientData[iTermId].stockLevelData.thresh_hold >= 100 ||
Term.pClientData[iTermId].stockLevelData.thresh_hold < 0 )
    {
    ErrorMessage(pECB, ERR_STOCKLEVEL_THRESHOLD_RANGE, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    return;
    }
    iRc = SQLStockLevel(pECB, iTermId, iSyncId,
Term.pClientData[iTermId].dbproc,
&Term.pClientData[iTermId].stockLevelData, iDeadlockRetry);
#ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if ( bConnectionPooling && iRc != -3 )
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
#endif
    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
        pEcbInfo->bFailed)
    return;
    if ( iRc )
    ErrorMessage(pECB, ERR_STOCKLEVEL_NOT_PROCESSED, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
    else
    WriteZString(pECB, MakeStockLevelForm(iTermId, iSyncId, FALSE) );
    return;
}
/* FUNCTION: int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA
*pNewOrderData)
 *
 * PURPOSE:This function extracts and validates the new order form data
from an http command string.
 *
 * ARGUMENTS:LPSTRlpszQueryStringclient browser http command string
```

```
 *  NEW_ORDER_DATA*pNewOrderDatapointer to new order data structure
 *
 * RETURNS:interror code indicating reason for failure
 *  ERR_SUCCESSnew order input data successfully parsed
 *
 *
 * COMMENTS:None
 *
 */
static int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA
*pNewOrderData)
{
    charszTmp[26];
    charszKey[26];
    inti;
    shortitems;
    BOOLbCheck;

    if ( !GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_FORM_MISSING_DID;
    if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_DISTRICT_INVALID;
    pNewOrderData->d_id = atoi(szTmp);
    if ( !GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_CUSTOMER_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_CUSTOMER_INVALID;
    pNewOrderData->c_id = atoi(szTmp);
    bCheck = FALSE;
    for(i=0, items=0; i<15; i++)
    {
    wsprintf(szKey, "IID%2.2d*", i);
    if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_MISSING_IID_KEY;
    if ( szTmp[0] )
    {
    //if blank lines between item ids
    if ( bCheck )
    return ERR_NEWORDER_ITEM_BLANK_LINES;
    if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_ITEMID_INVALID;
    pNewOrderData->Ol[i].ol_i_id = atoi(szTmp);

    wsprintf(szKey, "SP%2.2d*", i);
    if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_MISSING_SUPPW_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_SUPPW_INVALID;
    pNewOrderData->Ol[i].ol_supply_w_id = (short)atoi(szTmp);
    wsprintf(szKey, "Qty%2.2d*", i);
    if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_MISSING_QTY_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_QTY_INVALID;
    pNewOrderData->Ol[i].ol_quantity = atoi(szTmp);
    items++;
    if ( pNewOrderData->Ol[i].ol_i_id >= 1000000 || pNewOrderData-
>Ol[i].ol_i_id < 1 )
    return ERR_NEWORDER_ITEMID_RANGE;
    if ( pNewOrderData->Ol[i].ol_quantity >= 100 || pNewOrderData-
>Ol[i].ol_quantity < 1 )
    return ERR_NEWORDER_QTY_RANGE;
    }
    else
    {
    wsprintf(szKey, "SP%2.2d*", i);
    if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_MISSING_QTY_KEY;
    if ( szTmp[0] )
    return ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;
    wsprintf(szKey, "Qty%2.2d*", i);
    if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_MISSING_QTY_KEY;
    if ( szTmp[0] )
    return ERR_NEWORDER_QTY_WITHOUT_ITEMID;
    bCheck = TRUE;
    }
    }
    if ( items == 0 )
    return ERR_NEWORDER_NOITEMS_ENTERED;
    pNewOrderData->o_ol_cnt = items;
    return ERR_SUCCESS;
}
/* FUNCTION: int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA
*pPaymentData)
 *
 * PURPOSE:This function extracts and validates the payment form data
from an http command string.
 *
 * ARGUMENTS:LPSTRlpszQueryStringclient browser http command string
 *  PAYMENT_DATA*pPaymentDatapointer to payment data structure
 *
 * RETURNS:interror code indicating reason for failure
 *  ERR_SUCCESSall input data successfully parsed
```

```
 *
 * COMMENTS:None
 *
 */
static int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA
*pPaymentData)
{
    charszTmp[26];
    char*ptr;

    if ( !GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_DID_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_PAYMENT_DISTRICT_INVALID;
    pPaymentData->d_id = atoi(szTmp);
    if ( !GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_CID_KEY;
    if ( szTmp[0] && !IsNumeric(szTmp) )
    return ERR_PAYMENT_CUSTOMER_INVALID;
    pPaymentData->c_id = atoi(szTmp);
    if ( szTmp[0] == 0 )
    {
    if ( !GetKeyValue(lpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_CLT;
    _strupr( szTmp );
    strcpy(pPaymentData->c_last, szTmp);
    if ( strlen(pPaymentData->c_last) > 16 )
    return ERR_PAYMENT_LAST_NAME_TO_LONG;
    }
    else
    {
    if ( !GetKeyValue(lpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_CLT_KEY;
    if ( szTmp[0] )
    return ERR_PAYMENT_CID_AND_CLT;
    }

    if ( !GetKeyValue(lpszQueryString, "CDI*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_CDI_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_PAYMENT_CDI_INVALID;
    pPaymentData->c_d_id = atoi(szTmp);
    if ( !GetKeyValue(lpszQueryString, "CWI*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_CWI_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_PAYMENT_CWI_INVALID;
    pPaymentData->c_w_id = atoi(szTmp);

    if ( !GetKeyValue(lpszQueryString, "HAM*", szTmp, sizeof(szTmp)) )
    return ERR_PAYMENT_MISSING_HAM_KEY;
    ptr = szTmp;
    while( *ptr )
    {
    if ( *ptr == '.' )
    {
    ptr++;
    if ( !*ptr )
    break;
    if ( *ptr < '0' || *ptr > '9' )
    return ERR_PAYMENT_HAM_INVALID;
    ptr++;
    if ( !*ptr )
    break;
    if ( *ptr < '0' || *ptr > '9' )
    return ERR_PAYMENT_HAM_INVALID;
    if ( !*ptr )
    return ERR_PAYMENT_HAM_INVALID;
    }
    else if ( *ptr < '0' || *ptr > '9' )
    return ERR_PAYMENT_HAM_INVALID;
    ptr++;
    }
    pPaymentData->h_amount = atof(szTmp);
    if ( pPaymentData->h_amount >= 10000.00 || pPaymentData->h_amount <
0 )
    return ERR_PAYMENT_HAM_RANGE;
    return ERR_SUCCESS;
}
/* FUNCTION: int GetOrderStatusData(LPSTR lpszQueryString,
ORDER_STATUS_DATA *pOrderStatusData)
 *
 * PURPOSE:This function extracts and validates the payment form data
from an http command string.
 *
 * ARGUMENTS:LPSTRlpszQueryStringclient browser http command string
 *  ORDER_STATUS_DATA*pOrderStatusDatapointer to order status data
structure
 *
 * RETURNS:interror code indicating reason for failure
 *  ERR_SUCCESSsuccessfully parsed all required input data
 *
 * COMMENTS:None
 *
 */
static int GetOrderStatusData(LPSTR lpszQueryString, ORDER_STATUS_DATA
*pOrderStatusData)
```

```
{
    charszTmp[26];

    if ( !GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
    return ERR_ORDERSTATUS_MISSING_DID_KEY;
    if ( !IsNumeric(szTmp) )
    return ERR_ORDERSTATUS_DID_INVALID;
    pOrderStatusData->d_id = atoi(szTmp);
    if ( !GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)) )
    return ERR_ORDERSTATUS_MISSING_CID_KEY;
    if ( szTmp[0] == 0 )
    {
    pOrderStatusData->c_id = 0;
    if ( !GetKeyValue(lpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
    return ERR_ORDERSTATUS_MISSING_CLT_KEY;
    _strupr( szTmp );
    strcpy(pOrderStatusData->c_last, szTmp);
    if ( strlen(pOrderStatusData->c_last) > 16 )
    return ERR_ORDERSTATUS_CLT_RANGE;
    }
    else
    {
    if ( !IsNumeric(szTmp) )
    return ERR_ORDERSTATUS_CID_INVALID;
    pOrderStatusData->c_id = atoi(szTmp);
    if ( !GetKeyValue(lpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
    return ERR_ORDERSTATUS_MISSING_CLT_KEY;
    if ( szTmp[0] )
    return ERR_ORDERSTATUS_CID_AND_CLT;
    }
    return ERR_SUCCESS;
}
/* FUNCTION: BOOL ReadRegistrySettings(void)
 *
 * PURPOSE:This function reads the NT registry for startup parameters.
There parameters are
 *  under the TPCC key.
 *
 * ARGUMENTS:   None
 *
 * RETURNS:None
 *
 * COMMENTS:This function also sets up required operation variables to
their default value
 *  so if registry is not setup the default values will be used.
 *
 */
static BOOL ReadRegistrySettings(void)
{
    HKEYhKey;
    DWORDsize;
    DWORDtype;
    charszTmp[256];
    bLog= FALSE;
    iMaxWareHouses= 500;
    iThreads= 5;
    iQSlotts= 3000;
    iDelayMs= 100;
    iDeadlockRetry= (short)3;
    strcpy(szTpccLogPath, "tpcclog.");
#ifdef USE_ODBC
    bConnectionPooling = FALSE;
#endif
    if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\TPCC",
0, KEY_READ, &hKey) != ERROR_SUCCESS )
    return TRUE;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    {
    strcpy(szTpccLogPath, szTmp);
    strcat(szTpccLogPath, "tpcclog.");
    strcpy(szErrorLogPath, szTmp);
    strcat(szErrorLogPath, "tpccerr.");
    }
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    {
    if ( !stricmp(szTmp, "ON") )
    bLog = TRUE;
    }
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type, szTmp,
&size) == ERROR_SUCCESS )
    {
    iMaxWareHouses = atoi(szTmp);
    if ( iMaxWareHouses == 0 )
    iMaxWareHouses = 500;
    }
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "NumberOfDeliveryThreads", 0, &type,
szTmp, &size) == ERROR_SUCCESS )
    iThreads = atoi(szTmp);
    if ( !iThreads )
```

```
    iThreads = 5;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "QueueSlotts", 0, &type, szTmp, &size) ==
ERROR_SUCCESS )
    iQSlotts = atoi(szTmp);
    if ( !iQSlotts )
    iQSlotts = 3000;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "BackoffDelay", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iDelayMs = atoi(szTmp);
    if ( !iDelayMs )
    iDelayMs = 100;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iDeadlockRetry = (short)atoi(szTmp);
    if ( !iDeadlockRetry )
    iDeadlockRetry = (short)3;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp, &size)
== ERROR_SUCCESS )
    iMaxConnections = (short)atoi(szTmp);
    if ( !iMaxConnections )
    iMaxConnections = (short)25;
#ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "ConnectionPooling", 0, &type, szTmp,
&size) == ERROR_SUCCESS )
    if ( !stricmp(szTmp, "ON") )
    bConnectionPooling = TRUE;
    iConnectDelay = 500;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "ConnectionPoolRetryTime", 0, &type,
szTmp, &size) == ERROR_SUCCESS )
    iConnectDelay = atoi(szTmp);
    if ( !iConnectDelay )
    iConnectDelay = 500;
    #endif
#endif
    RegCloseKey(hKey);
    return FALSE;
}
/* FUNCTION: BOOL PostDeliveryInfo(short w_id, short o_carrier_id)
 *
 * PURPOSE:This function writes the delivery information to the delivery
pipe. The information is
 *  sent as a long.
 *
 * ARGUMENTS:shortw_idwarehouse id
 *  shorto_carrier_idcarrier id
 *
 * RETURNS:BOOLFALSEdelivery information posted successfully
 *  TRUEerror cannot post delivery info
 *
 * COMMENTS:The pipe is initially created with 16K buffer size this
should allow for
 *  up to 4096 deliveries to be queued before an overflow condition would
 *  occur. The only reason that an overflow would occur is if the
delivery
 *  application stopped listening while deliveries were being posted.
 *
 */
static BOOL PostDeliveryInfo(short w_id, short o_carrier_id)
{
    DELIVERY_TRANSACTIONdeliveryTransaction;
    intd;
    inti;
    GetLocalTime(&deliveryTransaction.queue);
    deliveryTransaction.w_id=w_id;
    deliveryTransaction.o_carrier_id=o_carrier_id;
    for(i=0; i<4; i++)
    {
    if ( WriteFile(hPipe, &deliveryTransaction,
sizeof(deliveryTransaction), &d, NULL) )
    return FALSE;
    if ( GetLastError() != ERROR_PIPE_BUSY )//ERROR_PIPE_LISTENING
    return TRUE;
    }
    return TRUE;
}
/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE:This function determines if a string is numeric. It fails if
any characters other
 *  than numeric and null terminator are present.
 *
 * ARGUMENTS:char*ptrpointer to string to check.
 *
 * RETURNS:BOOLFALSEif string is not all numeric
 *  TRUEif string contains only numeric characters i.e. '0' - '9'
 *
 * COMMENTS:None
 *
```

```
 */
static BOOL IsNumeric(char *ptr)
{
    if ( *ptr == 0 )
    return FALSE;
    while( *ptr && isdigit(*ptr) )
    ptr++;
    return ( !*ptr );
}
/* FUNCTION: void FormatHTMLString(char *szBuff, int iLen, char *szStr)
 *
 * PURPOSE:This function Handles translation of HTML specific character
field data
 *  when an HTML output form is generated.
 *
 * ARGUMENTS:char*szBuffReturned string information
 *  char*szStrinput string to be formatted.
 *  intiLenLength of returned string
 *
 * RETURNS:none
 *
 * COMMENTS:The length paramter is the absolute length of the returned
string in
 *  HTML characters. For example the input string > would be returned as
 *  &gt; which would be counted as 1 character.If the number of input
 *  characters is less than the iLen parameter spaces are appended to
 *  the end of the string to ensure that at least iLen characters are
 *  returned in the szBuff parameter.
 *
 */
static void FormatHTMLString(char *szBuff, char *szStr, int iLen)
{
    while( iLen && *szStr )
    {
    switch( *szStr )
    {
    case '>':
    *szBuff++ = '&';
    *szBuff++ = 'g';
    *szBuff++ = 't';
    *szBuff++ = ';';
    szStr++;
    break;
    case '<':
    *szBuff++ = '&';
    *szBuff++ = 'l';
    *szBuff++ = 't';
    *szBuff++ = ';';
    szStr++;
    break;
    case '&':
    *szBuff++ = '&';
    *szBuff++ = 'a';
    *szBuff++ = 'm';
    *szBuff++ = 'p';
    *szBuff++ = ';';
    szStr++;
    break;
    case '\"':
    *szBuff++ = '&';
    *szBuff++ = 'q';
    *szBuff++ = 'u';
    *szBuff++ = 'o';
    *szBuff++ = 't';
    *szBuff++ = ';';
    szStr++;
    break;
    default:
    *szBuff++ = *szStr++;
    break;
    }
    iLen--;
    }
    while( iLen-- )
    *szBuff++ = ' ';
    *szBuff = 0;
    return;
}
```

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) Console Application" 0x0103
# TARGTYPE "Win32 (x86) External Target" 0x0106
!IF "$(CFG)" == ""
CFG=tux_client - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to tux_client - Win32
Debug.
!ENDIF
!IF "$(CFG)" != "tpcc - Win32 Release" && "$(CFG)" != "tpcc - Win32
Debug" &&\
 "$(CFG)" != "tpcc_tux - Win32 Release" && "$(CFG)" != "tpcc_tux - Win32
Debug"\
 && "$(CFG)" != "tux_server - Win32 Release" && "$(CFG)" !=\
 "tux_server - Win32 Debug" && "$(CFG)" != "tux_client - Win32 Release"
&&\
 "$(CFG)" != "tux_client - Win32 Debug"
!MESSAGE Invalid configuration "$(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="tux_client - Win32 Debug"
!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 "tpcc_tux - Win32 Release" (based on "Win32 (x86) External
Target")
!MESSAGE "tpcc_tux - 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 "tux_client - Win32 Release" (based on\
 "Win32 (x86) Console Application")
!MESSAGE "tux_client - Win32 Debug" (based on\
 "Win32 (x86) Console Application")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF
!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
################################################################################
#######
# Begin Project
# PROP Target_Last_Scanned "tux_client - Win32 Debug"
!IF  "$(CFG)" == "tpcc - Win32 Release"
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "tpcc\Release"
# PROP BASE Intermediate_Dir "tpcc\Release"
# PROP BASE Target_Dir "tpcc"
# PROP BASE Cmd_Line "NMAKE /f tpcc.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tpcc\tpcc.exe"
# PROP BASE Bsc_Name "tpcc\tpcc.bsc"
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "tpcc\Release"
# PROP Intermediate_Dir "tpcc\Release"
# PROP Target_Dir "tpcc"
# PROP Cmd_Line "NMAKE /f tpcc.mak CFG=Release"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tpcc\tpcc.exe"
# PROP Bsc_Name "tpcc\tpcc.bsc"
OUTDIR=.\tpcc\Release
INTDIR=.\tpcc\Release
ALL :
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 "tpcc\Debug"
# PROP BASE Intermediate_Dir "tpcc\Debug"
# PROP BASE Target_Dir "tpcc"
# PROP BASE Cmd_Line "NMAKE /f tpcc.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tpcc\tpcc.exe"
# PROP BASE Bsc_Name "tpcc\tpcc.bsc"
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "tpcc\Debug"
# PROP Intermediate_Dir "tpcc\Debug"
# PROP Target_Dir "tpcc"
# PROP Cmd_Line "NMAKE /f tpcc.mak CFG=Debug"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tpcc\tpcc.exe"
# PROP Bsc_Name "tpcc\tpcc.bsc"
OUTDIR=.\tpcc\Debug
```

```
INTDIR=.\tpcc\Debug
ALL :
CLEAN :
    -@erase
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Release"
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "tpcc_tux\Release"
# PROP BASE Intermediate_Dir "tpcc_tux\Release"
# PROP BASE Target_Dir "tpcc_tux"
# PROP BASE Cmd_Line "NMAKE /f tpcc_tux.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tpcc_tux\tpcc_tux.exe"
# PROP BASE Bsc_Name "tpcc_tux\tpcc_tux.bsc"
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "tpcc_tux\Release"
# PROP Intermediate_Dir "tpcc_tux\Release"
# PROP Target_Dir "tpcc_tux"
# PROP Cmd_Line "NMAKE /f tpcc_tux.mak CFG=Release"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tpcc_tux\tpcc_tux.exe"
# PROP Bsc_Name "tpcc_tux\tpcc_tux.bsc"
OUTDIR=.\tpcc_tux\Release
INTDIR=.\tpcc_tux\Release
ALL :
CLEAN :
    -@erase
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "tpcc_tux\Debug"
# PROP BASE Intermediate_Dir "tpcc_tux\Debug"
# PROP BASE Target_Dir "tpcc_tux"
# PROP BASE Cmd_Line "NMAKE /f tpcc_tux.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tpcc_tux\tpcc_tux.exe"
# PROP BASE Bsc_Name "tpcc_tux\tpcc_tux.bsc"
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "tpcc_tux\Debug"
# PROP Intermediate_Dir "tpcc_tux\Debug"
# PROP Target_Dir "tpcc_tux"
# PROP Cmd_Line "NMAKE /f tpcc_tux.mak CFG=Debug"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tpcc_tux\tpcc_tux.exe"
# PROP Bsc_Name "tpcc_tux\tpcc_tux.bsc"
```

```
OUTDIR=.\tpcc_tux\Debug
INTDIR=.\tpcc_tux\Debug
ALL :
CLEAN :
    -@erase
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Release"
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "tux_server\Release"
# PROP BASE Intermediate_Dir "tux_server\Release"
# PROP BASE Target_Dir "tux_server"
# PROP BASE Cmd_Line "NMAKE /f tux_server.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tux_server\tux_server.exe"
# PROP BASE Bsc_Name "tux_server\tux_server.bsc"
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "tux_server\Release"
# PROP Intermediate_Dir "tux_server\Release"
# PROP Target_Dir "tux_server"
# PROP Cmd_Line "NMAKE /f tux_server.mak CFG=Release"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tux_server\tux_server.exe"
# PROP Bsc_Name "tux_server\tux_server.bsc"
OUTDIR=.\tux_server\Release
INTDIR=.\tux_server\Release
ALL :
CLEAN :
    -@erase
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "tux_server\Debug"
# PROP BASE Intermediate_Dir "tux_server\Debug"
# PROP BASE Target_Dir "tux_server"
# PROP BASE Cmd_Line "NMAKE /f tux_server.mak"
# PROP BASE Rebuild_Opt "/a"
# PROP BASE Target_File "tux_server\tux_server.exe"
# PROP BASE Bsc_Name "tux_server\tux_server.bsc"
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "tux_server\Debug"
# PROP Intermediate_Dir "tux_server\Debug"
# PROP Target_Dir "tux_server"
# PROP Cmd_Line "NMAKE /f tux_server.mak CFG=Debug"
# PROP Rebuild_Opt "/a"
# PROP Target_File "tux_server\tux_server.exe"
```

```
# PROP Bsc_Name "tux_server\tux_server.bsc"
OUTDIR=.\tux_server\Debug
INTDIR=.\tux_server\Debug
ALL :
CLEAN :
    -@erase
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
!ELSEIF  "$(CFG)" == "tux_client - Win32 Release"
# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "tux_client\Release"
# PROP BASE Intermediate_Dir "tux_client\Release"
# PROP BASE Target_Dir "tux_client"
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "tux_client\Release"
# PROP Intermediate_Dir "tux_client\Release"
# PROP Target_Dir "tux_client"
OUTDIR=.\tux_client\Release
INTDIR=.\tux_client\Release
ALL : "$(OUTDIR)\tux_client.exe"
CLEAN :
    -@erase "$(INTDIR)\getopt.obj"
    -@erase "$(INTDIR)\pipe_routines.obj"
    -@erase "$(INTDIR)\tux_client.obj"
    -@erase "$(INTDIR)\util.obj"
    -@erase "$(OUTDIR)\tux_client.exe"
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
CPP=cl.exe
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"
/YX /c
# ADD CPP /nologo /W3 /GX /O2 /I "c:\tuxedo\include" /I
".\..\sqlptk\include" /I "d:\tuxedo\include" /D "NDEBUG" /D "_CONSOLE"
/D "WIN32" /D "TUX" /YX /c
CPP_PROJ=/nologo /ML /W3 /GX /O2 /I "c:\tuxedo\include" /I\
 ".\..\sqlptk\include" /I "d:\tuxedo\include" /D "NDEBUG" /D "_CONSOLE"
/D\
 "WIN32" /D "TUX" /Fp"$(INTDIR)/tux_client.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\tux_client\Release/
CPP_SBRS=.\.
.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<
.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<
.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<
.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/tux_client.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib c:\tuxedo\lib\libtux.lib c:\tuxedo\lib\libbuft.lib
c:\tuxedo\lib\libtux2.lib c:\tuxedo\lib\libfml.lib
c:\tuxedo\lib\libfml32.lib c:\tuxedo\lib\libgp.lib
..\sqlptk\lib\ntwdblib.lib /nologo /subsystem:console /machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib c:\tuxedo\lib\libtux.lib c:\tuxedo\lib\libbuft.lib\
 c:\tuxedo\lib\libtux2.lib c:\tuxedo\lib\libfml.lib
c:\tuxedo\lib\libfml32.lib\
 c:\tuxedo\lib\libgp.lib ..\sqlptk\lib\ntwdblib.lib /nologo
/subsystem:console\
 /incremental:no /pdb:"$(OUTDIR)/tux_client.pdb" /machine:I386\
 /out:"$(OUTDIR)/tux_client.exe"
LINK32_OBJS= \
    "$(INTDIR)\getopt.obj" \
    "$(INTDIR)\pipe_routines.obj" \
    "$(INTDIR)\tux_client.obj" \
    "$(INTDIR)\util.obj"
"$(OUTDIR)\tux_client.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<
!ELSEIF  "$(CFG)" == "tux_client - Win32 Debug"
# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "tux_client\Debug"
```

```
# PROP BASE Intermediate_Dir "tux_client\Debug"
# PROP BASE Target_Dir "tux_client"
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "tux_client\Debug"
# PROP Intermediate_Dir "tux_client\Debug"
# PROP Target_Dir "tux_client"
OUTDIR=.\tux_client\Debug
INTDIR=.\tux_client\Debug
ALL : "$(OUTDIR)\tux_client.exe"
CLEAN :
    -@erase "$(INTDIR)\getopt.obj"
    -@erase "$(INTDIR)\pipe_routines.obj"
    -@erase "$(INTDIR)\tux_client.obj"
    -@erase "$(INTDIR)\util.obj"
    -@erase "$(INTDIR)\vc40.idb"
    -@erase "$(INTDIR)\vc40.pdb"
    -@erase "$(OUTDIR)\tux_client.exe"
    -@erase "$(OUTDIR)\tux_client.ilk"
    -@erase "$(OUTDIR)\tux_client.pdb"
"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"
CPP=cl.exe
# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE" /YX /c
# ADD CPP /nologo /W3 /Gm /GX /Zi /Od /I "c:\tuxedo\include" /I
".\..\sqlptk\include" /I "d:\tuxedo\include" /D "_DEBUG" /D "_CONSOLE"
/D "WIN32" /D "TUX" /YX /c
CPP_PROJ=/nologo /MLd /W3 /Gm /GX /Zi /Od /I "c:\tuxedo\include" /I\
 ".\..\sqlptk\include" /I "d:\tuxedo\include" /D "_DEBUG" /D "_CONSOLE"
/D\
 "WIN32" /D "TUX" /Fp"$(INTDIR)/tux_client.pch" /YX /Fo"$(INTDIR)/"\
 /Fd"$(INTDIR)/" /c
CPP_OBJS=.\tux_client\Debug/
CPP_SBRS=.\.
.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<
.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<
.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<
.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
RSC=rc.exe
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/tux_client.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /debug /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib c:\tuxedo\lib\libtux.lib c:\tuxedo\lib\libbuft.lib
c:\tuxedo\lib\libtux2.lib c:\tuxedo\lib\libfml.lib
c:\tuxedo\lib\libfml32.lib c:\tuxedo\lib\libgp.lib
..\sqlptk\lib\ntwdblib.lib /nologo /subsystem:console /debug
/machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib c:\tuxedo\lib\libtux.lib c:\tuxedo\lib\libbuft.lib\
 c:\tuxedo\lib\libtux2.lib c:\tuxedo\lib\libfml.lib
c:\tuxedo\lib\libfml32.lib\
 c:\tuxedo\lib\libgp.lib ..\sqlptk\lib\ntwdblib.lib /nologo
/subsystem:console\
 /incremental:yes /pdb:"$(OUTDIR)/tux_client.pdb" /debug /machine:I386\
 /out:"$(OUTDIR)/tux_client.exe"
LINK32_OBJS= \
    "$(INTDIR)\getopt.obj" \
    "$(INTDIR)\pipe_routines.obj" \
    "$(INTDIR)\tux_client.obj" \
    "$(INTDIR)\util.obj"
"$(OUTDIR)\tux_client.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<
!ENDIF
################################################################################
#######
# Begin Target
# Name "tpcc - Win32 Release"
# Name "tpcc - Win32 Debug"
!IF  "$(CFG)" == "tpcc - Win32 Release"
".\tpcc\tpcc.exe" :
   CD C:\users\mckee\src\win32\tpcc\dll\tpcc
   NMAKE /f tpcc.mak CFG=Release
!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
```

```
".\tpcc\tpcc.exe" :
   CD C:\users\mckee\src\win32\tpcc\dll\tpcc
   NMAKE /f tpcc.mak CFG=Debug
!ENDIF
################################################################################
#######
# Begin Source File
SOURCE=.\src\util.c
!IF  "$(CFG)" == "tpcc - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\pipe_routines.c
!IF  "$(CFG)" == "tpcc - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\sqlroutines.c
!IF  "$(CFG)" == "tpcc - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\TPCC.C
!IF  "$(CFG)" == "tpcc - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\error.c
!IF  "$(CFG)" == "tpcc - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"
!ENDIF
# End Source File
# End Target
################################################################################
#######
# Begin Target

# Name "tpcc_tux - Win32 Release"
# Name "tpcc_tux - Win32 Debug"
!IF  "$(CFG)" == "tpcc_tux - Win32 Release"
".\tpcc_tux\tpcc_tux.exe" :
   CD C:\users\mckee\src\win32\tpcc\dll\tpcc_tux
   NMAKE /f tpcc_tux.mak CFG=Release
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
".\tpcc_tux\tpcc_tux.exe" :
   CD C:\users\mckee\src\win32\tpcc\dll\tpcc_tux
   NMAKE /f tpcc_tux.mak CFG=Debug
!ENDIF
################################################################################
#######
# Begin Source File
SOURCE=.\src\util.c
!IF  "$(CFG)" == "tpcc_tux - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\pipe_routines.c
!IF  "$(CFG)" == "tpcc_tux - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\TPCC.C
!IF  "$(CFG)" == "tpcc_tux - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\tux_sql.c
!IF  "$(CFG)" == "tpcc_tux - Win32 Release"
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\error.c
!IF  "$(CFG)" == "tpcc_tux - Win32 Release"
```

```
!ELSEIF  "$(CFG)" == "tpcc_tux - Win32 Debug"
!ENDIF
# End Source File
# End Target
################################################################################
#######
# Begin Target
# Name "tux_server - Win32 Release"
# Name "tux_server - Win32 Debug"
!IF  "$(CFG)" == "tux_server - Win32 Release"
".\tux_server\tux_server.exe" :
   CD C:\users\mckee\src\win32\tpcc\dll\tux_server
   NMAKE /f tux_server.mak CFG=Release
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
".\tux_server\tux_server.exe" :
   CD C:\users\mckee\src\win32\tpcc\dll\tux_server
   NMAKE /f tux_server.mak CFG=Debug
!ENDIF
################################################################################
#######
# Begin Source File
SOURCE=.\src\tux_server.c
!IF  "$(CFG)" == "tux_server - Win32 Release"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\sqlroutines.c
!IF  "$(CFG)" == "tux_server - Win32 Release"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\error.c
!IF  "$(CFG)" == "tux_server - Win32 Release"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\tux_server\tux_server.mak
!IF  "$(CFG)" == "tux_server - Win32 Release"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
!ENDIF
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\util.c
!IF  "$(CFG)" == "tux_server - Win32 Release"
!ELSEIF  "$(CFG)" == "tux_server - Win32 Debug"
!ENDIF
# End Source File
# End Target
################################################################################
#######
# Begin Target
# Name "tux_client - Win32 Release"
# Name "tux_client - Win32 Debug"
!IF  "$(CFG)" == "tux_client - Win32 Release"
!ELSEIF  "$(CFG)" == "tux_client - Win32 Debug"
!ENDIF
################################################################################
#######
# Begin Source File
SOURCE=.\src\util.c
DEP_CPP_UTIL_=\
    ".\src\util.h"\

"$(INTDIR)\util.obj" : $(SOURCE) $(DEP_CPP_UTIL_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\tux_client.c
DEP_CPP_TUX_C=\
    ".\..\sqlptk\include\sqldb.h"\
    ".\..\sqlptk\include\sqlfront.h"\
    ".\src\pipe_routines.h"\
    ".\src\tpcc.h"\
    ".\src\tpcc_real.h"\
    ".\src\tpcc_tux.h"\
    ".\src\TRANS.H"\
    ".\src\tux.h"\
    "\tuxedo\include\atmi.h"\
    "\tuxedo\include\tmenv.h"\
    {$(INCLUDE)}"\sys\types.h"\

"$(INTDIR)\tux_client.obj" : $(SOURCE) $(DEP_CPP_TUX_C) "$(INTDIR)"
```

```
   $(CPP) $(CPP_PROJ) $(SOURCE)
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\pipe_routines.c
DEP_CPP_PIPE_=\
    ".\..\sqlptk\include\sqldb.h"\
    ".\..\sqlptk\include\sqlfront.h"\
    ".\src\pipe_routines.h"\
    ".\src\tpcc.h"\
    ".\src\tpcc_real.h"\
    ".\src\tpcc_tux.h"\
    ".\src\TRANS.H"\
    ".\src\tux.h"\

"$(INTDIR)\pipe_routines.obj" : $(SOURCE) $(DEP_CPP_PIPE_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)
# End Source File
################################################################################
#######
# Begin Source File
SOURCE=.\src\getopt.c
DEP_CPP_GETOP=\
    ".\src\getopt.h"\

"$(INTDIR)\getopt.obj" : $(SOURCE) $(DEP_CPP_GETOP) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)
# 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_VALUE101
#define _APS_NEXT_COMMAND_VALUE40001
#define _APS_NEXT_CONTROL_VALUE1000
#define _APS_NEXT_SYMED_VALUE101
//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 processed in any order.
#define WELCOME_FORM1//beginning form no term id assigned, form id
#define MAIN_MENU_FORM2//term id assigned main menu form id
#define NEW_ORDER_FORM3//new order form id
#define PAYMENT_FORM4//payment form id
#define DELIVERY_FORM5//delivery form id
#define ORDER_STATUS_FORM6//order status id
#define STOCK_LEVEL_FORM7//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 transactions
typedef struct _DELIVERY_TRANSACTION
{
    SYSTEMTIMEqueue;//time delivery transaction queued
    shortw_id;//delivery warehouse
    shorto_carrier_id;//carrier id
} DELIVERY_TRANSACTION;
#ifdef USE_ODBC
    typedef struct _DBPROCESS
    {
    HDBChdbc;
    HSTMThstmt;
    intspid;
    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
{
    intinUse;//in use flag allows client entries to be reused
    intw_id;//warehouse id assigned at welcome form
    intd_id;//district id assigned at welcome form
    PDBPROCESSdbproc;//dblib connection pointer
    intspid;//spid assigned from dblib
```

```
    intiSyncId;//syncronization id
    intiTickCount;//time of last access;
    intiTermId;//terminal id of http stream connection
    charszBuffer[4096];//form buffer each HTML form is built for a
client in here
    NEW_ORDER_DATAnewOrderData;//new order form data
    PAYMENT_DATApaymentData;//payment form data
    ORDER_STATUS_DATAorderStatusData;//order status form data
    DELIVERY_DATAdeliveryData;//delivery form data
    STOCK_LEVEL_DATAstockLevelData;//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
{
    intiAvailable;//total allocated terminal array entries
    intiNext;//next available terminal array element
    intiMasterSyncId;//syncronization id
    BOOLbInit;//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;
    #endif
```

../tpcc_tux/tpcc_tux.mak

```
    !IF "$(CFG)" == ""
    CFG=Debug
    !MESSAGE No configuration specified.  Defaulting to Debug
    !ENDIF
    !IF "$(SQL_LOC)" == ""
    SQL_LOC=..\..\sqlptk
    !MESSAGE No SQL_LOC specified.  Defaulting to C:\MSSQL\DBLIB
    !ENDIF
    !IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
    !MESSAGE Invalid configuration "$(CFG)" specified.
    !MESSAGE You can specify a configuration when running NMAKE on this
    makefile
    !MESSAGE by defining the macro CFG on the command line.  For example:
    !MESSAGE
    !MESSAGE NMAKE CFG="Debug"
    !MESSAGE
    !MESSAGE Possible choices for configuration are:
    !MESSAGE
    !MESSAGE "Release"
    !MESSAGE "Debug"
    !MESSAGE
    !ERROR An invalid configuration is specified.
    !ENDIF
    OUTDIR          = .
    SRCDIR          = ..\Src
    OBJDIR          = .\Objs.$(CFG)
    OUTDIR          = .\Bin
    ODBC= \progra~1\msdev
    DBLIB           = $(SQL_LOC)
    DBLIBINC        = $(DBLIB)\INCLUDE
    ODBCINCDIR= $(ODBC)\INCLUDE
    DBLIBDIR        = $(DBLIB)\LIB
    ODBCLIBDIR= $(ODBC)\LIB
    !IF "$(CFG)" != "Debug"
    LDEBUG       =
    CDEBUG       =
    LDEBUG_RG    =
    CDEBUG_RG    =
    DEBUG        =
    FLAGS        = /D "WIN32" /D "_WINDOWS"
    OPT          = /Ot
    !ELSE
    LDEBUG       = /debug /pdb:$(OBJDIR)\tpcc1.pdb
    CDEBUG       = /Zi /Yd
    LDEBUG_RG    = /debug /pdb:$(OBJDIR)\install.pdb
    CDEBUG_RG    = /Zi /Yd /Fd$(OBJDIR)\install.pdb
    FLAGS        = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
    OPT          = /Od
    !ENDIF
    LINK32_LIBS1    = user32.lib msacm32.lib advapi32.lib
    $(DBLIBDIR)\ntwdblib.lib
    LINK32_OBJS1    = "$(OBJDIR)\tpcc1.obj" "$(OBJDIR)\tpcc1.res"
    "$(OBJDIR)\tux_sql.obj" "$(OBJDIR)\error.obj" "$(OBJDIR)\util.obj"
    "$(OBJDIR)\pipe_routines.obj"
    LINK32_DEF1     = "$(SRCDIR)\tpcc1.def"
    LINK32_FLAGS1   = /nologo /subsystem:windows /dll /incremental:no
    $(LDEBUG) /def:"$(LINK32_DEF1)" /out:"$(OBJDIR)\tpcc1.dll"
    LINK32_LIBS2    = user32.lib msacm32.lib advapi32.lib
    $(ODBCLIBDIR)\odbc32.LIB
    LINK32_OBJS2    = "$(OBJDIR)\tpcc2.obj" "$(OBJDIR)\tpcc2.res"
    LINK32_DEF2     = "$(SRCDIR)\tpcc2.def"
    LINK32_FLAGS2   = /nologo /subsystem:windows /dll /incremental:no
    $(LDEBUG) /def:"$(LINK32_DEF2)" /out:"$(OBJDIR)\tpcc2.dll"
    LINK32_LIBS_RG  = user32.lib gdi32.lib advapi32.lib version.lib
    comctl32.lib
    LINK32_OBJS_RG  = "$(OBJDIR)\install.obj" "$(OBJDIR)\install.res"
```

```
LINK32_FLAGS_RG = /nologo /subsystem:windows /incremental:no
$(LDEBUG_RG) /out:$(OUTDIR)\install.exe
ALL:    $(OBJDIR)\. $(OBJDIR)\tpcc1.dll
$(OBJDIR)\.:
    if not exist $(OBJDIR) md $(OBJDIR)
$(OUTDIR)\.:
    if not exist $(OUTDIR) md $(OUTDIR)
"$(OBJDIR)\tpcc1.obj":   "$(SRCDIR)\tpcc.c" "$(SRCDIR)\tpcc.h"
    cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(DBLIBINC) $(FLAGS)
/Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\tpcc1.obj /c "$(SRCDIR)\tpcc.c"
"$(OBJDIR)\tux_sql.obj":   "$(SRCDIR)\tux_sql.c" "$(SRCDIR)\tpcc.h"
    cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(DBLIBINC) $(FLAGS)
/Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\tux_sql.obj /c "$(SRCDIR)\tux_sql.c"
"$(OBJDIR)\pipe_routines.obj":   "$(SRCDIR)\pipe_routines.c"
"$(SRCDIR)\tpcc.h"
    cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(DBLIBINC) $(FLAGS)
/Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\pipe_routines.obj /c
"$(SRCDIR)\pipe_routines.c"
"$(OBJDIR)\error.obj":   "$(SRCDIR)\error.c" "$(SRCDIR)\error.h"
    cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(DBLIBINC) $(FLAGS)
/Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\error.obj /c "$(SRCDIR)\error.c"
"$(OBJDIR)\util.obj":   "$(SRCDIR)\util.c" "$(SRCDIR)\util.h"
    cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(DBLIBINC) $(FLAGS)
/Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\util.obj /c "$(SRCDIR)\util.c"
$(OBJDIR)\tpcc1.res:     $(SRCDIR)\tpcc1.rc
    rc.exe /l 0x409 /fo $(OBJDIR)\tpcc1.res $(FLAGS) $(SRCDIR)\tpcc1.rc
$(OBJDIR)\tpcc1.dll:     $(LINK32_OBJS1) $(LINK32_DEF1)
    link.exe   $(LINK32_FLAGS1) $(LINK32_OBJS1) $(LINK32_LIBS1)
"$(OBJDIR)\tpcc2.obj":   "$(SRCDIR)\tpcc.c" "$(SRCDIR)\tpcc.h"
    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"
$(OBJDIR)\tpcc2.res:     $(SRCDIR)\tpcc2.rc
    rc.exe /l 0x409 /fo $(OBJDIR)\tpcc2.res $(FLAGS) $(SRCDIR)\tpcc2.rc
$(OBJDIR)\tpcc2.dll:     $(LINK32_OBJS2) $(LINK32_DEF2)
    link.exe   $(LINK32_FLAGS2) $(LINK32_OBJS2) $(LINK32_LIBS2)
$(OBJDIR)\delisrv1.exe:  $(SRCDIR)\delisrv.c $(SRCDIR)\delisrv.h
    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
$(OBJDIR)\delisrv2.exe:  $(SRCDIR)\delisrv.c $(SRCDIR)\delisrv.h
    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
$(OBJDIR)\install.res:  $(SRCDIR)\install.rc $(OBJDIR)\tpcc1.dll
$(OBJDIR)\tpcc2.dll $(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
    rc.exe /l 0x409 /fo$(OBJDIR)\install.res /i $(OBJDIR) /i $(SRCDIR)
$(FLAGS) $(SRCDIR)\install.rc
$(OBJDIR)\install.obj:  $(SRCDIR)\install.c $(OBJDIR)\tpcc1.dll
$(OBJDIR)\tpcc2.dll $(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
$(OBJDIR)\install.res
    cl -W3 $(CDEBUG) /Fo$(OBJDIR)\install.obj /c $(SRCDIR)\install.c
$(OUTDIR)\install.exe:     $(OBJDIR)\install.obj $(OBJDIR)\install.res
    link.exe @<<
  $(LINK32_FLAGS_RG) $(LINK32_OBJS_RG) $(LINK32_LIBS_RG)
<<
```

Tpcc1.def

```
LIBRARY TPCC1.DLL
EXPORTS
    GetExtensionVersion@1
    HttpExtensionProc@2
```

Tpcc1.rc

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
////////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#ifndef _MAC
////////////////////////////////////////////////////////////////////////////
////
//
// Version
//
VS_VERSION_INFO VERSIONINFO
```

```
 FILEVERSION 0,4,0,0
 PRODUCTVERSION 0,4,0,0
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x2L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "Comments", "TPC-C HTML DLL Server (DBLIB)\0"
            VALUE "CompanyName", "Microsoft\0"
            VALUE "FileDescription", "TPC-C HTML DLL Server (DBLIB)\0"
            VALUE "FileVersion", "0, 4, 0, 0\0"
            VALUE "InternalName", "tpcc\0"
            VALUE "LegalCopyright", "Copyright © 1996\0"
            VALUE "OriginalFilename", "tpcc.dll\0"
            VALUE "ProductName", "Microsoft tpcc\0"
            VALUE "ProductVersion", "0, 4, 0, 0\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END
#endif    // !_MAC
#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////
////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////////////
////
#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
//////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```

Tpcc2.def

```
LIBRARY TPCC2.DLL
EXPORTS
    GetExtensionVersion@1
    HttpExtensionProc@2
```

Tpcc2.rc

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"
#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
//////////////////////////////////////////////////////////////////////////
////
#undef APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////
////
// English (U.S.) resources
```

```
#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#ifndef _MAC
/////////////////////////////////////////////////////////////////////////////
////
//
// Version
//
VS_VERSION_INFO VERSIONINFO
 FILEVERSION 0,4,0,0
 PRODUCTVERSION 0,4,0,0
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x2L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "Comments", "TPC-C HTML DLL Server (ODBC)\0"
            VALUE "CompanyName", "Microsoft\0"
            VALUE "FileDescription", "TPC-C HTML DLL Server (ODBC)\0"
            VALUE "FileVersion", "0, 4, 0, 0\0"
            VALUE "InternalName", "tpcc\0"
            VALUE "LegalCopyright", "Copyright © 1996\0"
            VALUE "OriginalFilename", "tpcc.dll\0"
            VALUE "ProductName", "Microsoft tpcc\0"
            VALUE "ProductVersion", "0, 4, 0, 0\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END
#endif    // !_MAC
#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// TEXTINCLUDE
//
1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END
2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END
3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END
#endif    // APSTUDIO_INVOKED
#endif    // English (U.S.) resources
/////////////////////////////////////////////////////////////////////////////
////
#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
////
//
// Generated from the TEXTINCLUDE 3 resource.
//
/////////////////////////////////////////////////////////////////////////////
////
#endif    // not APSTUDIO_INVOKED
```

trans.h

```
/*  FILE:TRANS.H
 *  Microsoft TPC-C Kit Ver. 3.00.000
 *  Audited 08/23/96By Francois Raab
 *  PURPOSE:Header file for ISAPI TPCC.DLL, defines structures and
functions used in the isapi tpcc.dll.
 *
 *  Copyright Microsoft inc. 1996, All Rights Reserved
 *
 *  Author:PhilipDu, from tpcc.h by DamienL
 *  DamienL@Microsoft.com
 *  philipdu@Microsoft.com
```

```
 */
#ifndef _INC_TRANS
    #define _INC_TRANS
    #ifdef USE_ODBC
    #ifndef TIMESTAMP_STRUCT
    #include <sqltypes.h>
    #include <sql.h>
    #include <sqlext.h>
    #endif
    #else
    #ifndef _INC_SQLFRONT
    #define DBNTWIN32
    #include <sqlfront.h>
    #include <sqldb.h>
    #endif
    #endif
    #ifndef DBINT
    typedef long DBINT;
    #endif
    #defineDEFCLPACKSIZE4096
    #defineDEADLOCKWAIT10
    // 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;
shorto_commit_flag;
#ifdef USE_ODBC
TIMESTAMP_STRUCTo_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_STRUCTol_delivery_d;
#else
DBDATEREC ol_delivery_d;
#endif
} OL_ORDER_STATUS_DATA;
typedef struct
{
short               w_id;
short               d_id;
long                c_id;
char                c_first[FIRST_NAME_LEN+1];
char                c_middle[MIDDLE_NAME_LEN+1];
char                c_last[LAST_NAME_LEN+1];
double              c_balance;
long                o_id;
#ifdef USE_ODBC
TIMESTAMP_STRUCT o_entry_d;
#else
DBDATEREC           o_entry_d;
#endif
short               o_carrier_id;
OL_ORDER_STATUS_DATA OlOrderStatusData[MAX_OL_ORDER_STATUS_ITEMS];
short               o_ol_cnt;
long                num_deadlocks;
char     execution_status[STATUS_LEN];
} ORDER_STATUS_DATA;
typedef struct
{
long   o_id;
} DEL_ITEM;
typedef struct
{
short       w_id;
short       o_carrier_id;
SYSTEMTIME    queue_time;
long        num_deadlocks;
DEL_ITEM  DelItems[10];
char        execution_status[STATUS_LEN];
} DELIVERY_DATA;
typedef struct
{
short w_id;
short d_id;
short thresh_hold;
long  low_stock;
long  num_deadlocks;
char  execution_status[STATUS_LEN];
} STOCK_LEVEL_DATA;
#endif
```

tux.h

```
#ifndef TUX_H_INCLUDED
#define TUX_H_INCLUDED
#define SERVICE_CHARS 32
typedef union
{
    NEW_ORDER_DATANewOrderData;
    PAYMENT_DATAPaymentData;
    ORDER_STATUS_DATAOrderStatusData;
    DELIVERY_DATADeliveryData;
    STOCK_LEVEL_DATAStockLevelData;
    charErrorMsg[4096]; // ack!!
} TRANS_DATA;
typedef struct
{
    int TermId;
    int SyncId;
    int bDeadlock;
    int bFailed;
    short DeadlockRetry;
    int Error;
    int Return;
    // Note: Trans must be last
    TRANS_DATA Trans;
} TUX_DATA;
typedef struct
{
    char Service[SERVICE_CHARS];
    // Note: Data must be last
    TUX_DATA Data;
} TUX_MSG;
// macros to compute the size of various bits of TUX_MSG.  It is
// not enough to just add up the fields because of possible alignment
// issues
#define MSG_HEADER_SIZE(p)((DWORD)(((char *)&(p)->Data.Trans) - ((char
*)(p))))
#define NEW_ORDER_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(NEW_ORDER_DATA))
#define PAYMENT_SIZE(p) ((MSG_HEADER_SIZE((p)) + sizeof(PAYMENT_DATA))
#define ORDER_STATUS_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(ORDER_STATUS_DATA))
#define DELIVERY_SIZE(p) ((MSG_HEADER_SIZE((p)) + sizeof(DELIVERY_DATA))
#define STOCK_LEVEL_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(STOCK_LEVEL_DATA))
#endif
```

tux_client.c

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <direct.h>
#include "atmi.h"               /* TUXEDO  Header File */
#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 "trans.h"
#include "tpcc.h"
#include "pipe_routines.h"
#include "tux.h"
#define SERVICE_BUF_SIZE 16
typedef char * EXTENSION_CONROL_BLOCK;
const int TIMEOUT=1000*30;   // timeout in milliseconds
const int ARGSIZE=1024;
const char *LOG_PATH="c:\\temp\\tux_logs\\";
const char *LOG_NAME="client_%d.txt";
// Global variables set as parameters
int InitialCreate=0;
int ClientNumber=0;
char *TuxBuffer;
BOOL TuxInit()
{
    BOOL bReturn = FALSE;
    if (tpinit((TPINIT *) NULL) == -1)
    fprintf(stderr, "tpinit failed\n");
    else
    {
    TuxBuffer = (char *) tpalloc("CARRAY", NULL, sizeof(TUX_MSG));
    if (TuxBuffer != NULL)
    bReturn = TRUE;
    else
    {
```

```
    fprintf(stderr, "tpalloc of buffer failed\n");
    tpterm();
    }
    }
    return bReturn;
}
void TuxCleanup(void)
{
    tpfree(TuxBuffer);
    tpterm();
}
BOOL TuxTransaction(char *Service, void *Data, long BufSize, long
*pnRead)
{
    memcpy(TuxBuffer, Data, BufSize);
#ifdef _DEBUG
    fprintf(stderr,"about to tpcall Service %s, bufsize=%d\n", Service,
BufSize);
#endif
    if (tpcall(Service, TuxBuffer, BufSize, &TuxBuffer, pnRead,
TPNOTIME) == -1)
    {
    extern int tperrno;
    fprintf(stderr, "TuxTransaction: tpcall failed, tperrno=%d\n",
tperrno);
    return FALSE;
    }
#ifdef _DEBUG
    fprintf(stderr, "tp call retuned %d bytes\n", *pnRead);
#endif
    if (*pnRead < BufSize)
    {
    fprintf(stderr, "TuxTransaction: nRead(%d) < BufSize(%d)\n",
*pnRead, BufSize);
    return FALSE;
    }
    memcpy(Data, TuxBuffer, *pnRead);
    return TRUE;
}
void
HandleTransactions(HANDLE hPipe)
{
    TUX_MSG msg;
    DWORD nRead;
    HANDLE hEvent;

    hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
    if (hEvent == INVALID_HANDLE_VALUE)
    {
    fprintf(stderr, "Unable to create event handle\n");
    return;
    }
    while(ReadPipe(hPipe, hEvent, &msg, sizeof(msg), &nRead))
    {
    DWORD nWritten;
    if (!TuxTransaction(msg.Service, &msg.Data, nRead, &nRead))
    {
    fprintf(stderr, "TuxTransaction failed\n");
    break;
    }
    if (!WritePipe(hPipe, hEvent, &msg, nRead, &nWritten))
    {
    fprintf(stderr,"WritePipe Failed in HandleTransactions()\n");
    break;
    }
    if (nWritten != nRead)
    {
    fprintf(stderr, "HandleTransactions: nWritten(%d) != nRead(%d)\n",
    nWritten, nRead);
    }
    }
    CloseHandle(hEvent);
}
BOOL
StartAnother(char *name, int number, int InitialCreate)
{
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    char args[1024];
    sprintf(args, "%s -n %d %d", name, number, InitialCreate);
    memset(&si, 0, sizeof(si) );
    si.cb = sizeof(si);
    // Start the child process.
    if( !CreateProcess( NULL, // No module name (use command line).
        args,    // Command line.
        NULL,             // Process handle not inheritable.
        NULL,             // Thread handle not inheritable.
        FALSE,            // Set handle inheritance to FALSE.
        0,                // No creation flags.
        NULL,             // Use parent's environment block.
        NULL,             // Use parent's starting directory.
        &si,              // Pointer to STARTUPINFO structure.
        &pi )             // Pointer to PROCESS_INFORMATION structure.
    )
    {
```

```
    fprintf(stderr, "Unable to start another, number=%d\n",number);
    return FALSE;
    }
    return TRUE;
}
void
Usage(char *ProgName)
{
    fprintf(stderr,"usage: %s <initial create>\n", ProgName);
}
BOOL
Parse(int argc, char **argv)
{
    int c;
    BOOL bReturn=TRUE;
    extern char *optarg;
    extern int optind, optopt;
    while(bReturn && ((c = getopt(argc, argv, "n:")) != -1 ))
    {
    switch(c)
    {
    case 'n':
    ClientNumber = atoi(optarg);
    if (ClientNumber <=0)
    bReturn = FALSE;
    break;
    case ':':
    fprintf(stderr, "option %c requires an argument\n", optopt);
    bReturn = FALSE;
    break;
    case '?':
    bReturn = FALSE;
    break;
    default:
    // should not happen
    fprintf(stderr, "Parse in default case.\n");
    bReturn = FALSE;
    break;
    }
    }
    // See if we have any arguments left
    switch (argc - optind)
    {
    case 1:
    InitialCreate = atoi(argv[optind]);
    if (InitialCreate < 0)
    {
    bReturn = FALSE;
    break;
    }
    // fall through
    case 0:
    // nothing else specified - OK
    break;
    default:
    fprintf(stderr,"only one <initial_create> allowed\n");
    bReturn = FALSE;
    break;
    }
    return bReturn;
}
void
SetUpStderr(void)
{
    char buf[_MAX_PATH];
    strcpy(buf, LOG_PATH);
    _mkdir(LOG_PATH);
    sprintf(buf+strlen(buf), LOG_NAME, ClientNumber);
    freopen(buf, "w", stderr);
    setbuf(stderr, NULL);
}
int
main(int argc, char **argv)
{
    HANDLE hPipe;
    if (!Parse(argc, argv))
    {
    Usage(argv[0]);
    exit(1);
    }
#ifdef _DEBUG
    fprintf(stderr, "client %d starting (as thread 0x%x)\n",
ClientNumber, GetCurrentThreadId());
#endif
    SetUpStderr();
    if (!TuxInit())
    {
    fprintf(stderr, "tuxinit failed\n");
    exit(1);
    }
    if (ClientNumber == 0)
    {
    int i;
#ifdef _DEBUG
```

```
        fprintf(stderr, "Doing initial create of %d\n", InitialCreate);
    #endif
        for (i=1;i<InitialCreate;i++)
        StartAnother(argv[0], i, InitialCreate);
        }
        hPipe = OpenServerPipe(ClientNumber, INFINITE);
        if (hPipe == INVALID_HANDLE_VALUE)
        fprintf(stderr, "OpenServerPipe failed, error=%d\n", GetLastError());
        {
        if (ClientNumber >= InitialCreate)
        StartAnother(argv[0], ClientNumber + 1, InitialCreate);
        HandleTransactions(hPipe);
        CloseHandle(hPipe);
        }
        TuxCleanup();
        return 0;
    }
```

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 szErrorLogPath[]="\\inetpub\\wwwroot\\err_tpcc_tux.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;
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);
    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];
    {
    int i;
    for(i=0;i<argc;i++)
    printf("argv[%d]=%s\n", i, argv[i]);
    }
    Log("starting the tuxedo TPCC server");
    if (gethostname(App, sizeof(App)))
    strcpy(App, "TPCC");
    if (!SQLInit())
```

```
    {
    Log("SQLInit failed");
    return -1;
    }
    if (getenv("SERVER"))
    Server = strdup(getenv("SERVER"));
    if (Server == NULL)
    {
    Log("SERVER Environment variable not set");
    return -1;
    }
    if (SQLOpenConnection(NULL, 0, 0, &pdbproc, Server, Database, User,
Password, App, &spId))
    {
    Log("SQLOpenconnection failed");
    SQLCleanup();
    return -1;
    }
    return 0;
}
void
tpsvrdone(void)
{
    Log("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 _DEBUG
    Log("Beginning NEW_ORDER transaction\n");
#endif
    memcpy(&data, rqst->data, size);
    data.Return = SQLNewOrder(NULL,  data.TermId, data.SyncId, pdbproc,
&data.Trans.NewOrderData, data.DeadlockRetry);
    data.bDeadlock = pECBInfo->bDeadlock;
    data.bFailed = pECBInfo->bFailed;
    if (data.Error)
    {
    size = sizeof(data);
    strcpy(data.Trans.ErrorMsg, Term.pClientData[0].szBuffer);
    }
    memcpy(rqst->data, &data, size);
#ifdef _DEBUG
    Log("Finished NEWORDER transaction, bFailed=%d\n", data.bFailed);
#endif
    tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
}
void
STOCK_LEVEL(TPSVCINFO *rqst)
{
    PECBINFO pECBInfo = SQLGetECB(pdbproc);
    int size = rqst->len;

    memcpy(&data, rqst->data, size);
#ifdef _DEBUG
    Log("Beginning STOCK_LEVEL transaction\n");
#endif
    data.Return = SQLStockLevel(NULL,  data.TermId, data.SyncId,
pdbproc, &data.Trans.StockLevelData, data.DeadlockRetry);
    data.bDeadlock = pECBInfo->bDeadlock;
    data.bFailed = pECBInfo->bFailed;
    if (data.Error)
    {
    size = sizeof(data);
    strcpy(data.Trans.ErrorMsg, Term.pClientData[0].szBuffer);
    }
    memcpy(rqst->data, &data, size);
#ifdef _DEBUG
    Log("Finished STOCK_LEVEL transaction, bFailed=%d\n", data.bFailed);
#endif
    tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
}
void
PAYMENT(TPSVCINFO *rqst)
{
    PECBINFO pECBInfo = SQLGetECB(pdbproc);
    int size = rqst->len;
    memcpy(&data, rqst->data, size);
#ifdef _DEBUG
    Log("Beginning PAYMENT transaction\n");
#endif
    data.Return = SQLPayment(NULL,  data.TermId, data.SyncId, pdbproc,
&data.Trans.PaymentData, data.DeadlockRetry);

    data.bDeadlock = pECBInfo->bDeadlock;
    data.bFailed = pECBInfo->bFailed;

    if (data.Error)
    {
```

```
    size = sizeof(data);
    strcpy(data.Trans.ErrorMsg, Term.pClientData[0].szBuffer);
    }
    memcpy(rqst->data, &data, size);
#ifdef _DEBUG
    Log("Finished PAYMENT transaction\n");
#endif
    tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
}
void
ORDER_STATUS(TPSVCINFO *rqst)
{
    PECBINFO pECBInfo = SQLGetECB(pdbproc);
    int size = rqst->len;
    memcpy(&data, rqst->data, size);
#ifdef _DEBUG
    Log("Beginning ORDER_STATUS transaction, rqst->len=%d\n", rqst->len);
#endif
    data.Return = SQLOrderStatus(NULL,  data.TermId, data.SyncId,
pdbproc, &data.Trans.OrderStatusData, data.DeadlockRetry);
    data.bDeadlock = pECBInfo->bDeadlock;
    data.bFailed = pECBInfo->bFailed;
    if (data.Error)
    {
    size = sizeof(data);
    strcpy(data.Trans.ErrorMsg, Term.pClientData[0].szBuffer);
    }
    memcpy(rqst->data, &data, size);
#ifdef _DEBUG
    Log("Finished ORDER_STATUS transaction\n");
#endif
    tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
}
```

../tux_server/tux_server.mak

```
!IF "$(CFG)" == ""
CFG=Debug
!MESSAGE No configuration specified.  Defaulting to Debug
!ENDIF
!IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
!MESSAGE Invalid configuration "$(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 CFG="Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "Release"
!MESSAGE "Debug"
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF
OUTDIR          = .
SRCDIR          = ..\Src
OBJDIR          = .\Objs.$(CFG)
OUTDIR          = .\Bin
TUX = c:\tuxedo
SQL = ..\..\sqlptk
TUXINCLUDE= $(TUX)\include
SQLINCLUDE= $(SQL)\include
!IF "$(CFG)" != "Debug"
LDEBUG       =
CDEBUG       =
LDEBUG_RG    =
CDEBUG_RG    =
DEBUG        =
FLAGS        = /D "WIN32" /D "_WINDOWS"
OPT          = /Ot
!ELSE
LDEBUG       = /debug /pdb:$(OBJDIR)\tux_server.pdb
CDEBUG       = /Zi /Yd
FLAGS        = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
OPT          = /Od
!ENDIF
OBJS    = "$(OBJDIR)\tux_server.obj" "$(OBJDIR)\sqlroutines.obj"
"$(OBJDIR)\error.obj" "$(OBJDIR)\util.obj"
FLAGS   = /D "TUX" /c /nologo /MD /W3 $(FLAGS) $(CDEBUG) $(OPT)
/I$(SQLINCLUDE) /Fd$(OBJDIR)\tux_server.pdb
ALL:    $(OBJDIR)\. $(OBJDIR)\tux_server.exe
$(OBJDIR)\.:
    if not exist $(OBJDIR) md $(OBJDIR)
"$(OBJDIR)\tux_server.obj":
    cl.exe $(FLAGS) /I$(TUXINCLUDE) /Fd$(OBJDIR)\tux_server.pdb
/Fo$(OBJDIR)\tux_server.obj /c "$(SRCDIR)\tux_server.c"
"$(OBJDIR)\sqlroutines.obj":
    cl.exe $(FLAGS) /Fo$(OBJDIR)\sqlroutines.obj
"$(SRCDIR)\sqlroutines.c"
"$(OBJDIR)\error.obj":
    cl.exe $(FLAGS) /Fo$(OBJDIR)\error.obj "$(SRCDIR)\error.c"
"$(OBJDIR)\util.obj":   "$(SRCDIR)\util.c" "$(SRCDIR)\util.h"
```

```
	cl.exe $(FLAGS) /Fo$(OBJDIR)\util.obj "$(SRCDIR)\util.c"
$(OBJDIR)\tux_server.exe:   $(OBJS)
	build.cmd "$(OBJDIR)"
```

tux_sql.c

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#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 "trans.h"
#include "httpext.h"
#include "tpcc.h"
#include "tux.h"
#include "sqlroutines.h"
#include "pipe_routines.h"
#include "util.h"
const int ARGSIZE=1024;
const int PIPE_BUF_SIZE=4096;
staticCRITICAL_SECTIONCriticalSection;
void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr);
typedef struct
{
    int ThreadNumber;
    HANDLE hPipe;
} THREAD_DATA;
/*
 * This file contains the tuxedo client side routines.  Basically, they
 * are stubs for the routines in sqlroutines.c, which are used by the
 * tuxedo server
 */
DWORD TlsIndex;
int  ThreadCount=0;
BOOL SQLThreadAttach(void)
{
    THREAD_DATA *pData;
#ifdef _DEBUG
    fprintf(stderr, "SQLThread attach starts\n");
#endif
    pData = (THREAD_DATA *)malloc(sizeof(THREAD_DATA));
    if (!pData)
    return FALSE;
    memset(pData, 0, sizeof(*pData));
    EnterCriticalSection(&CriticalSection);
    pData->ThreadNumber = ThreadCount++;
    LeaveCriticalSection(&CriticalSection);
    pData->hPipe = OpenClientPipe(pData->ThreadNumber);
    if (pData->hPipe == INVALID_HANDLE_VALUE)
    {
#ifdef _DEBUG
    fprintf(stderr, "SQLThreadattach failed for thread %d\n", pData-
>ThreadNumber);
#endif
    free(pData);
    return FALSE;
    }
    else
    TlsSetValue(TlsIndex, pData);
#ifdef _DEBUG
    fprintf(stderr, "SQLThread attach succeeds for thread %d\n", pData-
>ThreadNumber);
#endif
    return TRUE;
}
BOOL SQLThreadDetach(void)
{
    THREAD_DATA *pData = TlsGetValue(TlsIndex);
    if (pData)
    {
    CloseHandle(pData->hPipe);
    free(pData);
    }
    return TRUE;
}
BOOL SQLInit(void)
{
    // Perform one time initialization.  According to the comments in
tpcc.c, this will
    // be called once when the DLL is loaded.  We assume that is true,
and also that
    // the caller has protected the call with a critical section.
    InitializeCriticalSection(&CriticalSection);
    TlsIndex = TlsAlloc();
    if (TlsIndex == 0xffffffff)
```

```
    {
    MessageBox(NULL, "TlsAlloc failed", "Init", MB_OK | MB_ICONSTOP);
    return FALSE;
    }
#ifdef _DEBUG
    fprintf(stderr, "TlsIndex = %d\n", TlsIndex);
#endif
}
void SQLCleanup(void)
{
    TlsFree(TlsIndex);
    TlsIndex = 0xffffffff;
    DeleteCriticalSection(&CriticalSection);
}
BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS **dbproc, char *server, char *database, char *user,
char *password, char *app, int *spid)
{
    PECBINFOpEcbInfo;
    //set pECB data into dbproc
    pEcbInfo = (PECBINFO)malloc(sizeof(ECBINFO));
    pEcbInfo->bDeadlock = FALSE;
    pEcbInfo->pECB= pECB;
    pEcbInfo->iTermId= iTermId;
    pEcbInfo->iSyncId= iSyncId;
    *dbproc = (DBPROCESS *)pEcbInfo;
    return FALSE;
}
BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS *dbproc)
{
    return FALSE;
}
BOOL TuxTransaction(char *Service, EXTENSION_CONTROL_BLOCK*pECB,
    int TermId, int SyncId, DBPROCESS *dbproc, short DeadlockRetry, void
*Data, long BufSize)
{
    THREAD_DATA *pData;
    TUX_MSG msg;
    DWORD nBytes;
    PECBINFO pECBInfo = (PECBINFO)dbproc; // forgive them them, for they
know not what they do...
    // we are pessimistic here
    pECBInfo->bFailed = TRUE;
    pData = TlsGetValue(TlsIndex);
    if (pData == NULL)
    {
    if (!SQLThreadAttach())
    {
    fprintf(stderr, "TuxTransaction: unable to attach\n");
    return FALSE;
    }
    pData = TlsGetValue(TlsIndex);
    }
    // fill the struct to ship to tux
    strcpy(msg.Service, Service);
    msg.Data.TermId = TermId;
    msg.Data.SyncId = SyncId;
    msg.Data.DeadlockRetry = DeadlockRetry;
    msg.Data.Error = FALSE;
    memcpy(&msg.Data.Trans, Data, BufSize);
    if (!WritePipe(pData->hPipe, NULL, &msg,
MSG_HEADER_SIZE(&msg)+BufSize, &nBytes))
    {
    fprintf(stderr, "Tuxtransaction: WritePipe Failed\n");
    return FALSE;
    }
    if (nBytes != MSG_HEADER_SIZE(&msg)+BufSize)
    {
    fprintf(stderr, "Tuxtransaction: short write, size=%d, written=%d\n",
    MSG_HEADER_SIZE(&msg)+BufSize, nBytes);
    return FALSE;
    }
    if (!ReadPipe(pData->hPipe, NULL, &msg, sizeof(msg), &nBytes))
    {
    fprintf(stderr, "Tuxtransaction: ReadPipe Failed\n");
    return FALSE;
    }
    if (msg.Data.Error)
    {
#ifdef _DEBUG
    fprintf(stderr, "msg.Error set, ErrorMsg=%s\n",
msg.Data.Trans.ErrorMsg);
#endif
    WriteZString(pECB, msg.Data.Trans.ErrorMsg);
    }
    // patch things up so the upper levels don't know this went
    // through tux
    pECBInfo->iTermId = TermId;
    pECBInfo->iSyncId = SyncId;
    pECBInfo->bDeadlock = msg.Data.bDeadlock;
    pECBInfo->bFailed = msg.Data.bFailed;
#ifdef _DEBUG
    fprintf(stderr, "bFailed=%d\n", pECBInfo->bFailed);
#endif
```

```
    memcpy(Data, &msg.Data.Trans, BufSize);
    return msg.Data.Return;
}
BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
{
    long ReceiveLen = sizeof(STOCK_LEVEL_DATA);
    return TuxTransaction("STOCK_LEVEL", pECB, iTermId, iSyncId, dbproc,
deadlock_retry, pStockLevel, sizeof(*pStockLevel));
}
int SQLNewOrder(EXTENSION_CONTROL_BLOCK*pECB, int iTermId, int iSyncId,
DBPROCESS *dbproc, NEW_ORDER_DATA  *pNewOrder, short deadlock_retry)
{
    return TuxTransaction("NEW_ORDER", pECB, iTermId, iSyncId, dbproc,
deadlock_retry, pNewOrder, sizeof(*pNewOrder));
}
int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short deadlock_retry)
{
    return TuxTransaction("PAYMENT", pECB, iTermId, iSyncId, dbproc,
deadlock_retry, pPayment, sizeof(*pPayment));
}
int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
{
    return TuxTransaction("ORDER_STATUS", pECB, iTermId, iSyncId,
dbproc, deadlock_retry, pOrderStatus, sizeof(*pOrderStatus));
}
PECBINFO SQLGetECB(PDBPROCESS p)
{
    return (PECBINFO)p;
}
```

util.c

```
#include <windows.h>
#include <string.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*pDestdestination string pointer
 *  char*pSrcsource string pointer
 *  intnnumber 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;
}
```

util.h

```
#ifndef TPCC_UTIL_H
#define TPCC_UTIL_H
void UtilStrCpy(char * pDest, char * pSrc, int n);
BOOL IsValidTermId(int TermId);
#endif
```

A.2 Driver

driver/generate.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 97/02/20 11:45:59 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include <stdio.h>
#include <values.h>
#include <unistd.h>
#include <time.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <fcntl.h>
#include <signal.h>
#include <math.h>
#include "shm_lookup.h"
#include "random.h"
#include <time.h>
```

```
int CLAST_CONST_C = 208;
int CID_CONST_C =  37;
int IID_CONST_C = 75;
int trans_type = 0;   /* type of transaction 0 == all */
extern ID warehouse;
extern ID district;
extern int no_warehouse;
extern int no_item;
extern int no_dist_pw;
extern int no_cust_pd;
extern int no_ord_pd;
extern int no_new_pd;
extern int tpcc_load_seed;
neworder_gen(t)
   neworder_trans *t;
   {
   int i;
   t->W_ID = warehouse;
   t->D_ID = RandomNumber(1, no_dist_pw);
   t->C_ID = NURandomNumber( 1023, 1, no_cust_pd, CID_CONST_C);
   t->O_OL_CNT = RandomNumber(5, 15);
   for (i=0; i<t->O_OL_CNT; i++)
      {
      t->item[i].OL_I_ID = NURandomNumber(8191, 1, no_item, IID_CONST_C);
      t->item[i].OL_SUPPLY_W_ID = RandomWarehouse(warehouse, scale, 1);
      t->item[i].OL_QUANTITY = RandomNumber(1, 10);
      }
   /* 1% of transactions roll back. Give the last order line a bad item */
   if (RandomNumber(1, 100) == 1)
      t->item[t->O_OL_CNT - 1].OL_I_ID = -1;
   }
payment_gen(t)
   payment_trans *t;
   {
   /* home warehouse is fixed */
   t->W_ID = warehouse;
   /* Random district */
   t->D_ID = RandomNumber(1, no_dist_pw);
   /* Customer is from remote warehouse and district 15% of the time */
   t->C_W_ID = RandomWarehouse(warehouse, scale, 15);
   if (t->C_W_ID == t->W_ID)
      t->C_D_ID = t->D_ID;
   else
      t->C_D_ID = RandomNumber(1, no_dist_pw);
   /* by name 60% of the time */
   t->byname = RandomNumber(1, 100) <= 60;
   if (t->byname)
      LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
       t->C_LAST);
   else
      t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);
   /* amount is random from [1.00..5,000.00] */
   t->H_AMOUNT = RandomNumber(100, 500000);
   }
ordstat_gen(t)
   ordstat_trans *t;
   {
   /* home warehouse is fixed */
   t->W_ID = warehouse;
   /* district is randomly selected from warehouse */
   t->D_ID = RandomNumber(1, no_dist_pw);

   /* by name 60% of the time */
   t->byname = RandomNumber(1, 100) <= 60;
   if (t->byname)
      LastName(NURandomNumber(255, 0, no_cust_pd/3 - 1, CLAST_CONST_C),
       t->C_LAST);
   else
      t->C_ID = NURandomNumber(1023, 1, no_cust_pd, CID_CONST_C);
   }
delivery_gen(t)
   delivery_trans *t;
   {
   t->W_ID = warehouse;
   t->O_CARRIER_ID = RandomNumber(1,10);
   }
stocklev_gen(t)
   stocklev_trans *t;
   {
   t->W_ID = warehouse;
   t->D_ID = district;
   t->threshold = RandomNumber(10, 20);
   }
int get_trans_type()
/*****************************************************************************
 * get_trans_type selects a transaction according to the weighted average
 * For TPC-C rev 3.0 and less and TPC-C rev 3.2 this is:
 *       new-order :  ???
 *       payment   : 43.0%
 *       order stat:  4.0%
 *       delivery  :  4.0%
 *       stock     :  4.0%
 *****************************************************************************/
   {
```

```
    static double weight[] = { 0.0,  0.0, .4305, .0405, .0405, .0405};
    double drand48();
    int type;
    double r;
    /* choose a random number between 0.0 and 1.0 */
    if (trans_type == 0) {
       r = drand48();

       /*
        * select one of STOCKLEV, DELIVERY, ORDSTAT and PAYMENT
        * based on weight
        */
       for (type = STOCKLEV;  type > NEWORDER;  type--) {
          r -= weight[type];
          if (r < 0) break;
       }
    } else {
       /* user wants only a certain type (say all stocklevel) so do that
          instead */
       type = trans_type;
    }
    /* return the value of the selected card, or NEWORDER if none selected */
    return type;
    }
```

lib/date.c

```
    /*************************************************************************
    ******
     @(#) Version: A.10.10 $Date: 96/04/02 16:26:09 $
     (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
    *************************************************************************
    *****/
    #include "tpcc.h"
    #include <time.h>
    /* macro to get starting day of a particular year (1901 thru 2100) */
    #define YEAR(yr)  ( (yr-1900)*365 + (yr-1900-1)/4 )
    CurrentDate(date)
    /************************************************************************
    CurrentDate fetches the current date and time
    ************************************************************************/
        DATE *date;
        {
        struct timeval time;
        struct timezone tz;
        /* get the current time of day */
        if (gettimeofday(&time, &tz) < 0)
        syserror("Can't get time of day\n");
        /* adjust the time of day by the timezone */
        time.tv_sec -= tz.tz_minuteswest * 60;
        /* convert seconds and days since EPOCH (Jan 1, 1970) */
        date->day = time.tv_sec / (24*60*60);
        date->sec = time.tv_sec - date->day * (24*60*60);
        /* convert to days since Jan 1, 1900 */
        date->day += YEAR(1970);
        }
    EmptyDate(date)
    /************************************************
    Get a NULL date and time
    ************************************************/
        DATE *date;
        {
        date->day = 0;   /* Use EMPTYNUM instead */
        date->sec = 0;
        }
    int IsEmptyDate(date)
        DATE *date;
        {
        return (date->day == 0 & date->sec == 0);
        }
    #define Feb29 (31+29-1)
    fmt_date(str, date)
    /************************************************************************
    fmt_date formats the DATE into a string  MM-DD-YY HH-MM-SS
    ************************************************************************/
        char str[20];
        DATE *date;
        {
        /* Note: should probably do date and time separately */
        int quad, year, month, day;
        int hour, minute, sec;
        static int dur[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        static int first = YES;
        day = date->day;
        sec = date->sec;
        /* if NULL date, then return empty string */
        if (day == EMPTY_NUM || sec == EMPTY_NUM)
        {str[0] = '\0';  return;}
        /* 2100, 1900 are NOT leap years.  If we are Feb 29 or later, add a
    day */
        if (day >= Feb29 + YEAR(2100))  day++;
        if (day >= Feb29)               day++;
        /* figure out which quad and day within quad we are in */
```

```
    quad = day / (4*365+1);
    day = day - quad * (4*365+1);
    /* get our year within quad and day within the year */
    if      (day < 1*365+1)    {year = 0;}
    else if (day < 2*365+1)    {year = 1; day -= 1*365+1;}
    else if (day < 3*365+1)    {year = 2; day -= 2*365+1;}
    else                       {year = 3; day -= 3*365+1;}
    /* if this is a leap year, february has 29 days */
    if (year == 0)          dur[1] = 29;
    else                    dur[1] = 28;
    /* decide which day and month we are */
    for (month = 0; day >= dur[month]; month++)
        day -= dur[month];
    /* decide what time of day it is */
    minute = sec / 60;
    sec = sec - minute * 60;
    hour = minute / 60;
    minute = minute - hour * 60;
    /* format the date and time */
    fmtint(str+0,  day+1, 2, ' ');
    str[2]='-';
    fmtint(str+3, month+1, 2, '0');
    str[5]='-';
    fmtint(str+6, 1900+quad*4+year, 4, '0');
    str[10] = ' ';
    fmtint(str+11, hour, 2, ' ');
    str[13] = ':';
    fmtint(str+14, minute, 2, '0');
    str[16] = ':';
    fmtint(str+17, sec, 2, '0');
    str[19] = '\0';
    }
```

lib/errlog.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 96/06/11 10:46:41 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include<fcntl.h>
#include<stdio.h>
#include<unistd.h>
#include<errno.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
extern int errno;
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", strerror(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, getpid());
  /* 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));
  close(fd);
  exit(1);
  }
```

lib/fmt.c

```
/*************************************************************************
 @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*************************************************************************/
#include "tpcc.h"
#include "iobuf.h"
#include <math.h>    /* needed for ceil (VM) */
#include <strings.h>
/* formatting routines.  */
/* Note: Currently use integer routines to format and convert.  Need to
   modify the code for cases when integers don't work.  */
fmt_money(str, m, width)
    char *str;
    MONEY m;
    int width;
    {
    if (m == EMPTY_FLT)
        {
        memset(str, '_', width);
        str[width] = '\0';
        return;
        }
    /* format it as a number with a leading blank */
    *str = ' ';
    fmt_flt(str+1, m/100, width-1, 2);
    /* fill in a leading dollar */
    while (*(str+1) == ' ')
        str++;
    *str = '$';
    }
double cvt_money(str)
    char *str;
    {
    char temp[81], *t, *s;
```

```
    double cvt_flt(), f;
    /* skip leading and trailing blanks */
    cvt_text(str, temp);
    /* remove leading $ */
    if (*temp == '$')   t = temp + 1;
    else                t = temp;
    /* start scan at current character */
    s = t;
    /* allow leading minus sign */
    if (*s == '-')
        s++;
    /* allow leading digits */
    while (isdigit(*s))
        s++;
    /* allow decimal pt and two decimal digits */
    if (*s == '.')    s++;
    if (isdigit(*s))  s++;
    if (isdigit(*s))  s++;
    /* There should be no more characters */
    if (*s != '\0') return INVALID_FLT;
    /* convert the floating pt number */
    f = cvt_flt(t);
    if (f == EMPTY_FLT)        return EMPTY_FLT;
    else if (f == INVALID_FLT) return INVALID_FLT;
    else                       return rint(f*100);
    }
fmt_num(str, n, width)
    char str[];
    int n;
    int width;
    {
    /* mark the end of the string */
    str[width] = '\0';
    /* if empty number, return the empty field */
    if (n == EMPTY_NUM)
        memset(str, '_', width);
    /* otherwise, convert the integer */
    else
        fmtint(str, n, width, ' ');
    debug("fmt_num:  n=%d  str=%s\n", n, str);
    }
cvt_num(str)
    char str[];
    {
    char text[81];
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_NUM;
    else
        return cvtint(text);
    }

fmt_flt(str, x, width, dec)
/*****************************************************************************
fmt_flt converts a floating pt number to a string   "999999.9999"
*****************************************************************************/
    char *str;
    double x;
    int width;
    int dec;
    {
    int negative;
    int integer, fract;
    double absolute;
    static double pow10[] =
    {1., 10., 100., 1000., 10000., 100000., 1000000., 10000000., 100000000.};
    /* mark the end of string */
    str[width] = '\0';
    /* if empty value, make it be an empty field */
    if (x == EMPTY_FLT)
        {
        memset(str, '_', width);
        return;
        }
    absolute = (x < 0)? -x: x;
    /* separate into integer and fractional parts */
    integer = (int) absolute;
    fract   = (absolute - integer) * pow10[dec] + .5;

    /* let the integer portion contain the sign */
    if (x < 0) integer = -integer;

    /* Format integer and fraction separately */
    fmtint(str, integer, width-dec-1, ' ');
    str[width-dec-1] = '.';
    fmtint(str+width-dec, fract, dec, '0');
    }
double cvt_flt(str)
    char str[];
    {
    char text[81];
    char *t;
    double value;
    int div;
```

```
    int fract;
    int negative;
    int i;
    /* normalize the text */
    cvt_text(str, text);
    if (*text == '\0')
        return EMPTY_FLT;
    negative = NO;
    fract = NO;
    value = 0;
    div = 1.0;
    negative = (text[0] == '-');
    if (negative)    t = text+1;
    else             t = text;
    for (;  *t !=  '\0' ; t++)
        {
        if (*t == '.')
            if (fract)  return INVALID_FLT;
            else        fract = YES;
        else if (isdigit(*t))
            {
            value = value*10 + (int)*t - (int)'0';
            if (fract) div *= 10;
            }
        else
            return INVALID_FLT;
        }
    if (fract)
        value /= div;
    if (negative)
        value = -value;
    return value;
    }
fmt_text(s, text, width)
    char *s, *text;
    int width;
    {
    /* if an empty string, then all underscores */
    if (*text == '\0')
        for (; width > 0; width--)
            *s++  = '_';
    /* otherwise, blank fill it */
    else
        {
        /* copy the text into the new buffer */
        for ( ; *text != '\0';  width--)
            *s++ = *text++;
        /* fill in the rest with blanks */
        for (;  width > 0;  width--)
            *s++ = ' ';
        }
    /* and finally, terminate the string */
    *s = '\0';
    }
cvt_text(s, text)
    char *s;
    char *text;
    {
    char *lastnb;

    /* skip leading blanks and underscores */
    for (; *s == ' ' || *s == '_'; s++)
        ;
    /* copy the characters, keeping track of last blank or underscore */
    lastnb = text-1;
    for (; *s != '\0'; *text++ = *s++)
        if (*s != ' ' && *s != '_')
            lastnb = text;
    /* truncate the text string to last nonblank character */
    *(lastnb+1) = '\0';
    }

fmtint(field, value, size, fill)
/*************************************************************************
fmtint formats an integer value into a character field to make the integer
right-justified within the character field, padded with leading fill
characters (e.g. leading blanks if a blank is passed in for the fill argument
*************************************************************************/
    int value;
    char *field;
    int size;
    char fill;
    {
    int negative;
    int dividend;
    int remainder;
    char *p;
    /* create characters from right to left */
    p = field + size - 1;
    /* make note if this is a negative number */
    negative = value < 0;
    if (negative)
        value = -value;
    /* Case: Null field.  Can't do anything */
```

```
    if (p < field)
        ;
    /* Case: value is zero.  Print a leading '0' */
    else if (value == 0)
        *p-- = '0';
    /* Otherwise, convert each digit in turn */
    else do
        {
        dividend = value / 10;
        remainder = value - dividend * 10;
        value = dividend;
        *p-- = (char) (  (int)'0'  +  remainder  );
        } while (p >= field && value > 0);
    /* insert a minus sign if appropriate */
    if (negative && p >= field)
        *p-- = '-';
    /* fill in leading characters */
    while (p >= field)
        *p-- = fill;
    }
int cvtint(str)
/*****************************************************************
getint extracts an integer value from the given character field
(ex: turns the string "123" into the integer 123)
*****************************************************************/
    char *str;
    {
    int value;
    char c;
    int negative;
    debug("cvtint: str=%s\n", str);
    negative = (*str == '-');
    if (negative)  str++;
    /* convert the integer */
    for (value = 0; isdigit(*str); str++)
        value = value*10 + (int)(*str) - (int)'0';
    /* if any non-digit characters, error */
    if (*str != '\0')
        return INVALID_NUM;
    /* make negative if there was a minus sign */
    if (negative)
        value = -value;
    debug("cvtint: value=%d\n", value);
    return value;
    }
fmt_phone(str, phone)
    char str[20];
    char *phone;
    {
    /* copy phone number and insert dashes  999999-999-999-9999 */
    str[0] = phone[0];  str[1] = phone[1]; str[2] = phone[2];
    str[3] = phone[3];  str[4] = phone[4]; str[5] = phone[5];
    str[6] = '-';
    str[7] = phone[6];  str[8] = phone[7]; str[9] = phone[8];
    str[10] = '-';
    str[11] = phone[9]; str[12] = phone[10]; str[13] = phone[11];
    str[14] = '-';
    str[15] = phone[12]; str[16] = phone[13]; str[17] = phone[14];
    str[18] = phone[15];
    str[19] = '\0';
    }
fmt_zip(str,zip)
    char str[20];
    char *zip;
    {
    /* copy zip code and insert dashes 99999-9999 */
    str[0] = zip[0];  str[1] = zip[1];  str[2] = zip[2];
    str[3] = zip[3];  str[4] = zip[4];
    str[5] = '-';
    str[6] = zip[5]; str[7] = zip[6]; str[8] = zip[7];  str[9] = zip[8];
    str[10] = '\0';
    }
```

lib/iobuf.c

```
/*******************************************************************************
 @(#) Version: A.10.10 $Date: 96/04/02 16:26:25 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*******************************************************************************/
#define DECLARE_IO_BUFFERS
#include "iobuf.h"
#undef DECLARE_IO_BUFFERS
#include "tpcc.h"
#include <sys/errno.h>
extern int errno;
string(str)
    char str[];
    {
    for (; *str != '\0'; str++)
        pushc(*str);
    }
push(str, len)
    char *str;
```

```
    int len;
    {
    for (; len > 0; len --)
        pushc(*str++);
    }
display(scr)
    iobuf *scr;
    {
/* Note: if problems doing output, let the input routine detect it */
    char *p;
    int len;
    for (p = scr->beg; p < scr->end; p+=len)
        {
        len = write(1, p, scr->end - p);
        if (len <= 0) break;
        }
    }
input(scr)
    iobuf *scr;
    {
    int len;
    /* read in as many characters as are available */
    len = read(0, scr->end, scr->max - scr->end);
    /* if end of input, then pretend we read an END character */
    if (len == 0 || (len == -1 && errno == ECONNRESET))
        {
        *scr->end = EOF;
        len = 1;
        }
    /* Check for errors */
    else if (len == -1)
        syserror("input(scr): unable to read stdin\n");
    /* update the pointers to reflect the new data */
    scr->end += len;
    *scr->end='\0';  /* for debugging */
    }
getkey()
    {
    if (in_buf->cur == in_buf->end)
        {
        flush();
        reset(in_buf);
        input(in_buf);
        }
    return popc();
    }
```

lib/iobuf.h

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 96/08/06 19:33:00 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
/*********************************************************************
History
941220 LAN Added definition and initialization of the line_col[] array.
       This was needed for modifications made of client program to do
       block I/O using a WYSE terminal.
*********************************************************************/
/* structure for screen emulation */
typedef struct
    {
    int row;
    int col;
    char buf[25][81];
    } screen_t;
typedef struct {
    char *beg;
    char *end;   /* for output buffers */
    char *max;
    char *cur;   /* for input buffers */
    } iobuf;
/* Macro do define an I/O buffer of x characters, initialized to empty */
#define define_iobuf(name, size)                    \
char name/**/_data[size];                           \
iobuf name[1] = {{name/**/_data, name/**/_data,     \
                 name/**/_data+size, name/**/_data}}
#define reset(buf)  if (1) { \
    (buf)->cur = (buf)->end = (buf)->beg;           \
    *(buf)->beg = '\0';                             \
    } else (void)0
#define flush()  if(1) { \
        display(out_buf);    \
        reset(out_buf);     \
     } else (void)0
/* Standard I/O to and from in_buf and out_buf */
#ifdef DECLARE_IO_BUFFERS
define_iobuf(output_stuff, 4*1024);
define_iobuf(input_stuff, 1024);
iobuf *in_buf = input_stuff;
iobuf *out_buf = output_stuff;
#else
iobuf *in_buf;
iobuf *out_buf;
```

```
#endif
#define pushc(c) if (1) { \
    if (out_buf->end >= out_buf->max)   \
        error("out_buf overflow:  beg=0x%x end=%d  max=%d\n",  \
         out_buf->beg, out_buf->end-out_buf->beg,out_buf->max-out_buf->beg); \
    *(out_buf->end++) = (c);  \
    *(out_buf->end) = '\0';  /* debug */ \
    } else (void)0
#define popc()  \
    (*in_buf->cur++)

/* Standard characters used for screen control */
#define ENTER '\015'
#define TAB   '\t'
#define BACKTAB '\02'  /*  ^B  */
#define CNTRLC '\03'
#define BACKSPACE '\010'
#define BELL '\07'
#define BLANK ' '
#define UNDERLINE '_'
#define ESCAPE '\033'
/*#define EOF    ((char)-1) */
#define TRIGGER '\021'  /* dc1 */
```

lib/random.c

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 97/02/20 11:47:10 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
*****************************************************************************/
#include "tpcc.h"
#include "string.h"
#include "random.h"
double drand48();
char lastNames[1000][16];
char customerData1[10][301];
char customerData2[10][201];
char stockData1[10][27];
char stockData2[10][25];
char historyData1[10][13];
char historyData2[10][13];
char citystreetData1[10][11];
char citystreetData2[10][11];
char firstNameData1[10][9];
char firstNameData2[10][9];
char StockDistrict[10][25];
char phoneData[10][17];
void GenerateLastNames()
{
    int      i;
    char *name;
    static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                        "ESE", "ANTI", "CALLY", "ATION", "EING"};
    for(i = 0; i < 1000; i++) {
        name = &lastNames[i];
        strcpy(name, n[(i/100)%10]);
        strcat(name, n[(i/10) %10]);
        strcat(name, n[(i/1)  %10]);
    }
}
int MakeNumberString(min, max, num)
    int min;
    int max;
    TEXT num[];
    {
    static char digit[]="0123456789";
    int length;
    int i;
    length = RandomNumber(min, max);

    for (i=0;  i<length;  i++)
        num[i] = digit[RandomNumber(0,9)];
    num[length] = '\0';
    return length;
    }
ID RandomWarehouse(local, scale, percent)
    ID local;
    ID scale;
    int percent;   /* percent of remote transactions */
    {
    ID w_id;
    /* For the given percent of the time, pick the local warehouse */
    if (RandomNumber(1, 100) > percent || scale == 1)
        w_id = local;
    /* Otherwise, pick a non-local warehouse */
    else
        {
        w_id = RandomNumber(2, scale);
        if (w_id == local)
            w_id = 1;
        }
    return w_id;
    }
```

```
/* Initialize a table of Random strings for the stock-district
   field in the stock table.  We can use a table of 10 elements
   and select randomly from this table via rule 4.3.2.2 in
   the TPC-C spec */
void InitRandomStrings()
{
    int i;
    for (i=0; i < 10; i++) {
          MakeAlphaString(24,24,&StockDistrict[i]);
          MakeAlphaString(300,300,&customerData1[i]);
          MakeAlphaString(0,200,&customerData2[i]);
          MakeAlphaString(26,26,&stockData1[i]);
          MakeAlphaString(0,24,&stockData2[i]);
          MakeAlphaString(12,12,&historyData1[i]);
          MakeAlphaString(0,12, &historyData2[i]);
          MakeAlphaString(10,10,&citystreetData1[i]);
          MakeAlphaString(0 ,10,&citystreetData2[i]);
          MakeAlphaString(8,8,&firstNameData1[i]);
          MakeAlphaString(0,8,&firstNameData2[i]);
          MakeNumberString(16,16,&phoneData[i]);
       }
       GenerateLastNames();
}
int MakeAlphaString(min, max, str)
    int min;
    int max;
    TEXT str[];
    {
    static char character[] = "abcdefghijklmnopqrstuvwxyz";
    int length;
    int i;
    length = RandomNumber(min, max);
    for (i=0;  i<length;  i++) {
       /* NOTE:  we use sizeof(character)-2 because of the following:
          subtract 1 because we are numbering from 0 instead of 1 and
          subtract 1 because the sizeof(character) is 1 greater than
          the data in character because of the inivisible C string
          terminator at the end. */
       str[i] = character[RandomNumber(0, sizeof(character)-2)];
    }
    str[length] = '\0';
    return length;
    }
void RandomPermutation(perm, n)
    int perm[];
    int n;
    {
    int i, r, t;
    /* generate the identity permutation to start with */
    for (i=1; i<=n; i++)
       perm[i] = i;
    /* randomly shuffle the permutation */
    for (i=1; i<=n; i++)
       {
       r = RandomNumber(i, n);
       t = perm[i]; perm[i] = perm[r]; perm[r] = t;
       }
    }
void RandomDelay(mean, adjust)
/*********************************************************************
random_sleep sleeps according to the TPC specification
*********************************************************************/
    double mean;
    double adjust;
    {
    double secs;
    double exponential();
    secs = exponential(mean);

    delay(secs+adjust);
    }
double exponential(mean)
/*********************************************************************
exponential generates a reverse exponential distribution
*********************************************************************/
    double mean;
    {
    double x;
    double log();
    x = -log(1.0-drand48()) * mean;
    return x;
    }
void Randomize()
    {
    srand48(time(0)+getpid());
    }
```

lib/random.h

```
/*********************************************************************
******
 @(#) Version: A.10.10 $Date: 97/02/20 11:47:20 $
 (c) Copyright 1996, Hewlett-Packard Company, all rights reserved.
```

```
*************************************************************************
*****/
#ifndef TPCC_RANDOM
#define TPCC_RANDOM
double drand48();
extern int    MakeNumberString();
extern ID     RandomWarehouse();
extern int    MakeAlphaString();
extern void   RandomPermutation();
extern void   RandomDelay();
extern double exponential();
extern void   Randomize();
extern char lastNames[1000][16];
extern char customerData1[10][301];
extern char customerData2[10][201];
extern char stockData1[10][27];
extern char stockData2[10][25];
extern char historyData1[10][13];
extern char historyData2[10][13];
extern char citystreetData1[10][11];
extern char citystreetData2[10][11];
extern char firstNameData1[10][9];
extern char firstNameData2[10][9];
extern char StockDistrict[10][25];
extern char phoneData[10][17];
/************************************************************************
*
RandomNumber selects a uniform random number from min to max inclusive
************************************************************************
*/
#define RandomNumber(min,max)  \
    ((int)(drand48() * ((int)(max) - (int)(min) + 1)) + (int)(min))
/************************************************************************
NURandomNumber selects a non-uniform random number
************************************************************************/
#define NURandomNumber(a, min, max, c) \
    ((RandomNumber(0, a) | RandomNumber(min, max)) + (c)) % \
    ((max) - (min) + 1) + (min)
#define SelectHistoryData(data) \
{ \
    strcpy(data,historyData1[RandomNumber(0,9)]);\
    strcat(data,historyData2[RandomNumber(0,9)]);\
}
#define SelectCityStreetData(data) \
{ \
    strcpy(data,citystreetData1[RandomNumber(0,9)]); \
    strcat(data,citystreetData2[RandomNumber(0,9)]); \
}
#define SelectFirstName(data) \
{ \
    strcpy(data,firstNameData1[RandomNumber(0,9)]); \
    strcat(data,firstNameData2[RandomNumber(0,9)]); \
}
#define SelectHistoryData(data) \
{ \
    strcpy(data,historyData1[RandomNumber(0,9)]); \
    strcat(data,historyData2[RandomNumber(0,9)]); \
}
#define SelectStockData(data) \
{ \
    strcpy(data,stockData1[RandomNumber(0,9)]); \
    strcat(data,stockData2[RandomNumber(0,9)]); \
}
#define SelectClientData(data) \
{ \
    strcpy(data,customerData1[RandomNumber(0,9)]); \
    strcat(data,customerData2[RandomNumber(0,9)]); \
}
#define SelectPhoneData(data) strcpy(data,phoneData[RandomNumber(0,9)])
#define SelectStockDistrict(data)
strcpy(data,StockDistrict[RandomNumber(0,9)])
#define MakeZip(zip) \
{ \
   MakeNumberString(4, 4, zip); \
   zip[4] = '1'; \
   zip[5] = '1'; \
   zip[6] = '1'; \
   zip[7] = '1'; \
   zip[8] = '1'; \
   zip[9] = '\0'; \
}
#define MakeAddress(str1, str2, city, state, zip) \
{ \
    SelectCityStreetData(str1); \
    SelectCityStreetData(str2); \
    SelectCityStreetData(city); \
    MakeAlphaString(2,2,state); \
    MakeZip(zip); \
}
#define LastName(num, name) strcpy(name, lastNames[(num)])
#define Original(str) \
{ \
    int len = strlen(str); \
    if (len >= 8) { \
```

```
        int pos = RandomNumber(0,(len-8)); \
        str[pos+0] = 'O'; \
        str[pos+1] = 'R'; \
        str[pos+2] = 'I'; \
        str[pos+3] = 'G'; \
        str[pos+4] = 'I'; \
        str[pos+5] = 'N'; \
        str[pos+6] = 'A'; \
        str[pos+7] = 'L'; \
    } \
}
#endif
```

Appendix B – Database Design

Build

diskinit.sql

```
    /*  TPC-C Benchmark Kit                                 */
    /*                                                      */
    /*  DISKINIT.SQL                                        */
    /*                                                      */
    /*  This script is used create the database devices     */
    use master
    go
    disk init name = "c_log1_dev",
        physname = "u:",
        vdevno   = 14,
        size     = 4096000
    go
    disk init name = "c_log2_dev",
        physname = "v:",
        vdevno   = 15,
        size     = 4096000
    go
    disk init name = "c_ordln1_dev",
        physname = "f:",
        vdevno   = 16,
        size     = 2340000
    go
    disk init name = "c_ordln2_dev",
        physname = "g:",
        vdevno   = 17,
        size     = 2340000
    go
    disk init name = "c_ordln3_dev",
        physname = "h:",
        vdevno   = 18,
        size     = 2340000
    go
    disk init name = "c_ordln4_dev",
        physname = "i:",
        vdevno   = 19,
        size     = 2340000
    go
    disk init name = "c_ordln5_dev",
        physname = "j:",
        vdevno   = 20,
        size     = 2340000
    go
    disk init name = "c_cs1_dev",
        physname = "k:",
        vdevno   = 21,
        size     = 4910000
    go
    disk init name = "c_cs2_dev",
        physname = "l:",
        vdevno   = 22,
        size     = 4910000
    go
    disk init name = "c_cs3_dev",
        physname = "m:",
        vdevno   = 23,
        size     = 4910000
    go
    disk init name = "c_cs4_dev",
        physname = "n:",
        vdevno   = 24,
        size     = 4910000
    go
    disk init name = "c_cs5_dev",
        physname = "o:",
        vdevno   = 25,
        size     = 4910000
    go
    disk init name = "c_misc1_dev",
        physname = "p:",
        vdevno   = 26,
        size     = 169000
    go
    disk init name = "c_misc2_dev",
        physname = "q:",
        vdevno   = 27,
        size     = 169000
    go
    disk init name = "c_misc3_dev",
        physname = "r:",
        vdevno   = 28,
        size     = 169000
    go
    disk init name = "c_misc4_dev",
        physname = "s:",
        vdevno   = 29,
```

```
    size     = 169000
go
disk init name = "c_misc5_dev",
    physname = "t:",
    vdevno   = 30,
    size     = 3449000
go
```

createdb.sql

```
/*  TPC-C Benchmark Kit                          */
/*                                               */
/*  CREATEDB.SQL                                 */
/*                                               */
/*  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   = 4550,
    c_ordln2_dev   = 4550,
    c_ordln3_dev   = 4550,
    c_ordln4_dev   = 4550,
    c_ordln5_dev   = 4550,
    c_cs1_dev    = 6940,
    c_cs2_dev    = 6940,
    c_cs3_dev    = 6940,
    c_cs4_dev    = 6940,
    c_cs5_dev    = 6940,
    c_cs1_dev    = 1890,
    c_cs2_dev    = 1890,
    c_cs3_dev    = 1890,
    c_cs4_dev    = 1890,
    c_cs5_dev    = 1890,
    c_cs1_dev    = 740,
    c_cs2_dev    = 740,
    c_cs3_dev    = 740,
    c_cs4_dev    = 740,
    c_cs5_dev    = 740,
    c_misc1_dev  = 328,
    c_misc2_dev  = 328,
    c_misc3_dev  = 328,
    c_misc4_dev  = 328,
    c_misc5_dev  = 6740
    log on c_log1_dev =   8000,
    c_log2_dev =  8000
for load
go
```

segment.sql

```
/*  TPC-C Benchmark Kit                                  */
/*                                                       */
/*  SEGMENT.SQL                                          */
/*                                                       */
/*  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_extendsegment misc_seg, c_misc3_dev
exec sp_extendsegment misc_seg, c_misc4_dev
exec sp_extendsegment misc_seg, c_misc5_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_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
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
            update stock set s_ytd        = s_ytd + @li_qty,
                             @s_quantity  = s_quantity,
                             s_quantity   = s_quantity - @li_qty +
                                 case when (s_quantity - @li_qty < 10)
then 91 else 0 end,
                             s_order_cnt  = s_order_cnt + 1,
                             s_remote_cnt = s_remote_cnt + case
                                 when (@li_s_w_id = @w_id) then 0 else 1
end,
                             @s_data      = s_data,
                             @s_dist      = case @d_id
                           when 1  then s_dist_01
                               when 2  then s_dist_02
                               when 3  then s_dist_03
                               when 4  then s_dist_04
                               when 5  then s_dist_05
                               when 6  then s_dist_06
                               when 7  then s_dist_07
                               when 8  then s_dist_08
                               when 9  then s_dist_09
                               when 10 then s_dist_10
                               end
                  where s_i_id = @li_id and
                 s_w_id = @li_s_w_id
            /* insert order_line data (using data from item and stock) */
                  insert into order_line values(@o_id,              /* from
district update */
                                                @d_id,      /* input
param          */
                                                @w_id,              /* input
param          */
                                                @li_no,             /*
orderline number     */
                                                @li_id,             /*
lineitem id          */
                                                @li_s_w_id,         /*
lineitem warehouse   */
                                                "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("%ORIGINAL%",@i_data) >
0) and
                                        (patindex("%ORIGINAL%",@s_data) > 0) )
                                then "B" else "G" end,
                           @i_price,
                           @i_price * @li_qty

               end
               else
           begin

               /*  no item found - triggers rollback condition */

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

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

```
    select @c_last     = c_last,
           @c_discount = c_discount,
           @c_credit   = c_credit,
       @c_id_local = c_id
    from customer holdlock
    where c_id   = @c_id and
          c_w_id = @w_id and
          c_d_id = @d_id
    /* insert fresh row into orders table */
    insert into orders values (@o_id,
                          @d_id,
                     @w_id,
              @c_id_local,
              @o_entry_d,
              0,
              @o_ol_cnt,
              @o_all_local)
    /* insert corresponding row into new-order table */
    insert into new_order values (@o_id,
      @d_id,
      @w_id)
    /*  select warehouse tax */
    select @w_tax = w_tax
    from warehouse holdlock
    where w_id = @w_id
    if (@commit_flag = 1)
    commit transaction n
    else
    /* all that work for nuthin!!! */
    rollback transaction n
    /* return order data to client */
    select @w_tax,
           @d_tax,
       @o_id,
       @c_last,
       @c_discount,
       @c_credit,
       @o_entry_d,
       @commit_flag
end
go
```

payment.sql

```
/*  File:
PAYMENT.SQL                                                 */
/*              Microsoft TPC-C Kit Ver.
3.00.000                            */
/*              Audited 08/23/96, By Francois
Raab                          */
/*
  */
/*              Copyright Microsoft,
1996                                  */
/*
  */
/*  Purpose:    Payment transaction for Microsoft TPC-C Benchmark
Kit      */
/*  Author:     Damien
Lindauer                                              */
/*
damienl@Microsoft.com                                          */
use tpcc
go
if exists (select name from sysobjects where name = "tpcc_payment" )
    drop procedure tpcc_payment
go
create proc tpcc_payment @w_id           smallint,
     @c_w_id         smallint,
     @h_amount       numeric(6,2),
     @d_id           tinyint,
     @c_d_id         tinyint,
     @c_id           int,
     @c_last         char(16) = ""


as
declare  @w_street_1     char(20),
         @w_street_2     char(20),
         @w_city         char(20),
         @w_state        char(2),
         @w_zip          char(9),
         @w_name         char(10),
         @d_street_1     char(20),
         @d_street_2     char(20),
         @d_city         char(20),
         @d_state        char(2),
         @d_zip          char(9),
         @d_name         char(10),
         @c_first        char(16),
         @c_middle       char(2),
         @c_street_1     char(20),
         @c_street_2     char(20),
         @c_city         char(20),
```

```
        @c_state        char(2),
        @c_zip          char(9),
        @c_phone        char(16),
        @c_since        datetime,
        @c_credit       char(2),
        @c_credit_lim   numeric(12,2),
        @c_balance      numeric(12,2),
        @c_discount     numeric(4,4),
        @data1          char(250),
        @data2          char(250),
        @c_data_1       char(250),
        @c_data_2       char(250),
        @datetime       datetime,
        @w_ytd          numeric(12,2),
        @d_ytd          numeric(12,2),
        @cnt            smallint,
        @val            smallint,
        @screen_data char(200),
     @d_id_local tinyint,
     @w_id_local smallint,
     @c_id_local int
select @screen_data = ""
begin tran p

    /* get payment date */
    select @datetime = getdate()

    if (@c_id = 0)
    begin
    /* get customer id and info using last name */
    select @cnt = count(*)
    from customer holdlock
    where c_last = @c_last and
          c_w_id = @c_w_id and
          c_d_id = @c_d_id

    select @val = (@cnt + 1) / 2
    set rowcount @val
    select @c_id = c_id
    from customer holdlock
    where c_last = @c_last and
          c_w_id = @c_w_id and
          c_d_id = @c_d_id
    order by c_w_id, c_d_id, c_last, c_first
    set rowcount 0
    end

/* get customer info and update balances */

update customer set
@c_balance     = c_balance = c_balance - @h_amount,
c_payment_cnt  = c_payment_cnt + 1,
c_ytd_payment  = c_ytd_payment + @h_amount,
@c_first       = c_first,
@c_middle      = c_middle,
    @c_last        = c_last,
    @c_street_1    = c_street_1,
@c_street_2    = c_street_2,
@c_city        = c_city,
@c_state       = c_state,
@c_zip         = c_zip,
@c_phone       = c_phone,
@c_credit      = c_credit,
@c_credit_lim  = c_credit_lim,
@c_discount    = c_discount,
@c_since       = c_since,
@data1         = c_data_1,
@data2         = c_data_2,
@c_id_local    = c_id
where c_id   = @c_id and
      c_w_id = @c_w_id and
      c_d_id = @c_d_id

/* if customer has bad credit get some more info */
if (@c_credit = "BC")
begin
/* compute new info */
select @c_data_2 = substring(@data1,209,42) +
   substring(@data2, 1, 208)
select @c_data_1 = convert(char(5),@c_id) +
   convert(char(4),@c_d_id) +
           convert(char(5),@c_w_id) +
           convert(char(4),@d_id) +
           convert(char(5),@w_id) +
           convert(char(19),@h_amount) +
           substring(@data1, 1, 208)
/* update customer info */
update customer set
       c_data_1 = @c_data_1,
   c_data_2 = @c_data_2
where c_id   = @c_id and
  c_w_id = @c_w_id and
      c_d_id = @c_d_id
select @screen_data = substring (@c_data_1,1,200)
```

```
    end



    /* get district data and update year-to-date */

    update district
    set d_ytd       = d_ytd + @h_amount,
    @d_street_1 = d_street_1,
    @d_street_2 = d_street_2,
    @d_city     = d_city,
    @d_state    = d_state,
    @d_zip      = d_zip,
    @d_name     = d_name,
    @d_id_local = d_id
    where d_w_id = @w_id and
          d_id   = @d_id
    /* get warehouse data and update year-to-date */
    update warehouse
    set w_ytd       = w_ytd + @h_amount,
    @w_street_1 = w_street_1,
        @w_street_2 = w_street_2,
        @w_city     = w_city,
        @w_state    = w_state,
        @w_zip      = w_zip,
        @w_name     = w_name,
    @w_id_local = w_id
    where w_id = @w_id

    /* create history record */

    insert into history values  (@c_id_local,
         @c_d_id,
     @c_w_id,
     @d_id_local,
     @w_id_local,
     @datetime,
     @h_amount,
     @w_name + "    " + @d_name)
commit tran p
/* return data to client */
select  @c_id,
    @c_last,
    @datetime,
    @w_street_1,
    @w_street_2,
    @w_city,
    @w_state,
    @w_zip,
    @d_street_1,
    @d_street_2,
    @d_city,
    @d_state,
    @d_zip,
    @c_first,
    @c_middle,
    @c_street_1,
    @c_street_2,
    @c_city,
    @c_state,
    @c_zip,
    @c_phone,
    @c_since,
    @c_credit,
    @c_credit_lim,
    @c_discount,
    @c_balance,
    @screen_data
go
```

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[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    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 defination
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_delivery_d);
    strcpy(ol_dist_info,orders_buf[i].o_ol[j].ol_dist_info);
    if (!bcp_sendrow(o_dbproc3))
    printf("Error, LoadOrderLineTable() failed calling bcp_sendrow().
Check error file.\n");
    order_line_rows_loaded++;
    // CheckForCommit(o_dbproc3, order_line_rows_loaded, "ORDER_LINE",
&order_line_time_start->time_start);
    }
    }
    bcp_batch(o_dbproc3);
    if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
    {
    bcp_done(o_dbproc3);
    dbclose(o_dbproc3);
    if (aptr->build_index == 1)
    BuildIndex("idxodlcl");
    }
    InterlockedIncrement(&order_threads_completed);
}
//============================================================================
//
// Function   : GetPermutation
//
//============================================================================
void GetPermutation(int perm[], int n)
{
    int i, r, t;
    for (i=1;i<=n;i++)
    perm[i] = i;
    for (i=1;i<=n;i++)
    {
    r = RandomNumber(i,n);
    t = perm[i];
    perm[i] = perm[r];
    perm[r] = t;
    }

}
//============================================================================
//
// Function   : CheckForCommit
//
//============================================================================
```

```
void CheckForCommit(DBPROCESS *dbproc,
        int rows_loaded,
    char *table_name,
        long *time_start)
{

    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
```

Microsoft Windows NT Version 4.0 Configuration Parameters

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

```
Computer Browser
Liscense Logging Service
Messenger
NT LM Security Support Provider
Remote Procedure Call (RPC) Service
Plug and Plan
Spooler
TCP/IP Netbios Helper
```

NT Registry

```
Infonet
Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
Class Name:        <NO CLASS>
Last Write Time:   6/24/97 - 12:21 PM
Value 0
  Name:            BandwidthLevel
  Type:            REG_DWORD
  Data:            0xffffffff

Value 1
  Name:            ListenBackLog
  Type:            REG_DWORD
  Data:            0x800

Value 2
  Name:            PoolThreadLimit
  Type:            REG_DWORD
  Data:            0x1fe

Value 3
  Name:            PoolThreadsLimit
  Type:            REG_DWORD
  Data:            0x1fe

Value 4
  Name:            ThreadTimeout
  Type:            REG_DWORD
  Data:            0x15180


Key Name:
SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\Filter
Class Name:        <NO CLASS>
Last Write Time:   5/21/97 - 4:23 PM
Value 0
  Name:            FilterType
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            NumDenySites
  Type:            REG_DWORD
  Data:            0

Value 2
  Name:            NumGrantSites
  Type:            REG_DWORD
  Data:            0


Key Name:
SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\MimeMap
Class Name:        <NO CLASS>
Last Write Time:   5/21/97 - 4:23 PM
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:
```

```
tpcc:
Key Name:          SOFTWARE\Microsoft\TPCC
Class Name:        <NO CLASS>
Last Write Time:   7/2/97 - 5:58 PM
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:            500

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:            3000

Value 7
  Name:            MaximumWarehouses
```

```
  Type:             REG_SZ
  Data:             900

Value 8
  Name:             NumberOfDeliveryThreads
  Type:             REG_SZ
  Data:             9

Value 9
  Name:             PATH
  Type:             REG_SZ
  Data:             C:\InetPub\wwwroot\

Value 10
  Name:             QueueSlots
  Type:             REG_SZ
  Data:             3000

Value 11
  Name:             QueueSlotts
  Type:             REG_SZ
  Data:             3000

w3svc:
Key Name:           SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
Class Name:         <NO CLASS>
Last Write Time:    7/2/97 - 5:58 PM
Value 0
  Name:             AcceptExOutstanding
  Type:             REG_DWORD
  Data:             0x800

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_NRDWBC1

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:             0x1c20

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:           0x186a0

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\Services\W3SVC\Parameters\Script Map
Class Name:        <NO CLASS>
Last Write Time:   5/21/97 - 4:23 PM
Value 0
  Name:            .idc
  Type:            REG_SZ
  Data:            C:\WINNT\System32\inetsrv\httpodbc.dll


Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots
Class Name:        <NO CLASS>
Last Write Time:   5/21/97 - 6:03 PM
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
```

Microsoft SQL Server Version 6.5 Startup Parameters

```
c:\mssql\binn\sqlservr -c -x -t1081 -t3502 -t812 -Cp3800 -Cd1410000
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
-t3052prints a message to the log at
    the start and end of each
    checkpoint
-t812omits sorting for write page
    ordering during checkpoints
-Cp3800specifies number of procedure
    cache buffers to allocate
-Cd1410000specifies number of data
    buffers to allocate
```

Microsoft SQL Server Version 6.5 Configuration Parameters

```
sp_configure:
1> 2> sp_configure
 name                                minimum     maximum     config_value
    run_value
 ----------------------------------- ----------- ----------- ------------
    -----------
 affinity mask                                 0  2147483647           15
             15
 allow updates                                 0           1            1
              1
```

```
 backup buffer size                      1
32            1
              1
 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          5000
           5000
 hash buckets                         4999
265003       265003
          265003
 language in cache                       3
100             3
              3
 LE threshold maximum                    2
500000           200
            200
 LE threshold minimum                    2
500000            20
             20
 LE threshold percent                    1
100             0
              0
 locks                                5000
2147483647          5000
           5000
 LogLRU buffers                          0
2147483647          1800
           1800
 logwrite sleep (ms)                    -1
500            -1
              -1
 max async IO                            1
1024            16
             16
 max lazywrite IO                        1
1024            64
             64
 max text repl size                      0
2147483647         65536
          65536
 max worker threads                     10
1024           130
            130
 media retention                         0
365             0
              0
 memory                                2800
1048576        900000
          900000
 nested triggers                         0
1              1
              1
 network packet size                   512
32767          2048
            2048
 open databases                          5
32767             8
              8
 open objects                          100
2147483647           200
            200
 priority boost                          0
1              0
              0
 procedure cache                         1
99             1
              1
 Protection cache size                   1
8192           15
             15
```

```
RA cache hit limit                      1         255           4
             4
RA cache miss limit                     1         255           3
             3
RA delay                                0         500          15
            15
RA pre-fetches                          1        1000           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           5
             5
time slice                             50        1000         100
           100
user connections                        5       32767         185
           185
user options                            0        4095           0
                      0
(1 row affected)
```

Server System Configuration Parameters

```
                              Board Information
HP NetServer LX System Board
         System
This system board contains the following on-board features:
        1)  2 CPU module slots supporting up to
                4 processors,
        2)  1 ECC Memory connectors
                (Up to 1.5GB Total Memory),
        3)  6 PCI connectors,
        4)  4 EISA connectors,
        5)  Static RAM Cache,
        6)  Shadow RAM Control,
        7)  Paged Memory,
        8)  Dual I/O APIC Units,
        9)  1 Parallel Port,
        10) 2 Serial Ports,
        11) Onboard Video Controller,
        12) Onboard Floppy Controller,
        13) Onboard IDE Hard Disk Controller,
        14) Onboard mouse/keyboard interface,
        15) 2 Onboard SCSI Controllers.

Manufacturer ................. Hewlett-Packard Co.
ID ........................... INT31C0
Category ..................... SYS
Board slot type .............. Embedded
Readable ID .................. Yes
Overlay name ................. INT31C0.OVL
Overlay version .............. 1.32
SYSTEMS GROUP
  System Identification and Version Information
    System Identification String ............... IDNOCD0L
   Config and Overlay Version .................. Overlay version: 1.20o

    BIOS Version String ........................ BIOS: 1.00.07.CD0L
     MP Spec Version: ........................... MP Spec V1.4 (Use for
NT
                                                              3.5x, 4.x)
```

```
   System Processor Modules
     CPU 1 in Slot 1 ........................... Pentium
(R) Pro  Intel CPU
                                                 at  200
MHz
     CPU 2 in Slot 1 ........................... Pentium
(R) Pro  Intel CPU
                                                 at  200
MHz
     CPU 1 in Slot 2 ........................... Pentium
(R) Pro  Intel CPU
                                                 at  200
MHz
     CPU 2 in Slot 2 ........................... Pentium
(R) Pro  Intel CPU
                                                 at  200
MHz


   System Processor Status
     CPU 1 in Slot 1 ........................... No
Failures Detected
     CPU 2 in Slot 1 ........................... No
Failures Detected
     CPU 1 in Slot 2 ........................... No
Failures Detected
     CPU 2 in Slot 2 ........................... No
Failures Detected
   System Performance
     Power-On Speed Option ..................... CPU
Speed=FAST
     APIC Interrupt Routing .................... Local
APIC, Mode A (Default)
     Secondary IOAPIC Control Option ........... Disable
Secondary IOAPIC
 MEMORY SUBSYSTEM GROUP
   Base Memory Options ......................... 640KB
Base Memory
   Options for Shadowing ROMs ................. Press
'Enter'to Modify the
                                                 Shadowing
Options
   Extended Memory Options (Cache, 1MB ISA Hole)   4063 MB
Extended Memory  /
                                                 512 KB
Cache (WB)
 On-Board Disk Controllers
   On-Board Floppy Controller ................. Enable
   On-board IDE Controller .................... Disable
 On-Board Communication Devices
   Serial Port 1 Configuration ................ Port 1
Disable
   Serial Port 2 Configuration ................ Port 2
Disable
   Parallel Port Configuration ................ Parallel
Port Disable
   Parallel Port Mode ......................... Parallel
Port Mode
                                                 ISA-
Compatible
   Parallel Port DMA (Valid only with ECP Mode) . No DMA
 FLOPPY DRIVE SUBSYSTEMS GROUP
   Floppy Drive A Options ..................... 3.5  inch
1.44/1.25 MB drive
   Floppy Drive B Options ..................... Disable
or Not Installed
 IDE SUBSYSTEM GROUP
   ISA IDE DMA Transfers ...................... Disable
   IDE Configuration -- Primary Master ........ No Drive
Detected
   IDE Drive Options -- Primary Master
     Multi-Sector Transfer .................... Auto
Configured
     Translation Mode ......................... Auto
Configured
     Fast Programmed I/O Modes ................ Auto
Configured
   IDE Configuration -- Primary Slave ......... No Drive
Detected
   IDE Drive Options -- Primary Slave
     Multi-Sector Transfer .................... Auto
Configured
     Translation Mode ......................... Auto
Configured
     Fast Programmed I/O Modes ................ Auto
Configured
   IDE Configuration -- Secondary Master ...... No Drive
Detected
   IDE Drive Options -- Secondary Master
     Multi-Sector Transfer .................... Auto
Configured
```

```
    Translation Mode ..... Auto Configured
    Fast Programmed I/O Modes ..... Auto Configured
  IDE Configuration -- Secondary Slave ..... No Drive Detected

  IDE Drive Options -- Secondary Slave
    Multi-Sector Transfer ..... Auto Configured
    Translation Mode ..... Auto Configured
    Fast Programmed I/O Modes ..... Auto Configured
BIOS MESSAGE LANGUAGE
  Language Support Options
   Current Language ..... English  (US)
KB and MOUSE SUBSYSTEM GROUP
  Keyboard and Mouse Options
    NumLock Options ..... OFF at Boot
    Typematic Speed ..... Auto
    Mouse Control Option ..... Mouse Auto detected
Console Redirection
  Console Redirection Control
    COM Port for Redirection ..... Disable
    Serial Port Baud Rate ..... 19.2K Baud
    Hardware Flow Control ..... None
    Select Terminal Type ..... ANSI
SECURITY SUBSYSTEMS GROUP
  Administrative Password Option ..... Disabled
  User Password Option ..... Disabled
  Hot Key Option ..... Disabled
  Lockout Timer  .....  10 Minutes
  Secure Boot Mode ..... Disable
  Video Blanking  ..... Disable
  Floppy Writes ..... Disable
BOOT SUBSYSTEM GROUP
  Boot Options
    First Boot Device ..... Boot Floppy
    Second Boot Device ..... Boot Hard Disk
  Boot From Embedded SCSI ..... Onboard Boot Priority
  Require User Interaction on POST Errors ..... Disable
SCSI ROM BIOS OPTIONS GROUP
  SCSI-A ROM BIOS Scan ..... Enable
  SCSI-B ROM BIOS Scan ..... Enable
LCD SUBSYSTEM GROUP
  LCD Display String Enable or Disable ..... Enable
  LCD Display String Before OS Boot ..... Default
MANAGEMENT SUBSYSTEM GROUP
  Temperature/Voltage Limit Control ..... Press 'Enter' to
modify the
                                          System Limits
  A/D Channel Enable Switch ..... Press 'Enter' to
                                          Enable/Disable the A/D
                                          Channels
  Speaker Options ..... Enable
  System Management Options

    System Management Mode ..... Enable
    Event Logging ..... Enable
    PCI System Error Detection ..... Disable
  Power Down Options
    Cycle Power on ASR ..... Disable
    Power Down on Critical Thermal or Voltage
    Condition ..... Disable
ADVANCED CONFIGURATION - !CAUTION!
  Advanced PCI Options
    PCI Master Latency Timer ..... 60h (Default)
    Bus Performance Mode ..... Auto Detect (Default)
    Address Bit Permuting ..... Enabled
    PCI Line Read Prefetch ..... Enabled (Default)
  Advanced CPU Options
    In-order Queue depth ..... Depth = 8
    Processor Retry Timer ..... Disabled
  Advanced Chipset Options
    GAT Mode ..... (Enable) Alias mode
(Default)
    Outbound Posting  ..... Enabled
    Memory Controller Page Open Policy ..... Hold page open with no
                                          requests
Reserved System Resources
  Reserve VGA Resources ..... Reserve VGA Memory (On-
board
                                          video)



                          Board Information
PCI Mass Storage Controller
        PCI 1
Manufacturer ..... PCI
ID ..... 101e9010
Category ..... MSD
Board slot type ..... PCI
Readable ID ..... No
Skirt ..... No
PCI Function 1 ..... Enabled


                          Board Information
PCI Mass Storage Controller
```

```
        PCI 2
Manufacturer ................. PCI
ID ........................... 101e9010
Category ..................... MSD
Board slot type .............. PCI
Readable ID .................. No
Skirt ........................ No
 PCI Function 1 ................................ Enabled


                    Board Information
PCI Mass Storage Controller
        PCI 3
Manufacturer ................. PCI
ID ........................... 101e9010
Category ..................... MSD
Board slot type .............. PCI
Readable ID .................. No
Skirt ........................ No
 PCI Function 1 ................................ Enabled


                    Board Information
PCI Mass Storage Controller
        PCI 4
Manufacturer ................. PCI
ID ........................... 101e9010
Category ..................... MSD
Board slot type .............. PCI
Readable ID .................. No
Skirt ........................ No
 PCI Function 1 ................................ Enabled


                    Board Information
PCI Mass Storage Controller
        PCI 5
Manufacturer ................. PCI
ID ........................... 101e9010
Category ..................... MSD
Board slot type .............. PCI
Readable ID .................. No
Skirt ........................ No
 PCI Function 1 ................................ Enabled


                    Board Information
PCI Ethernet Controller
        PCI 6
Manufacturer ................. PCI
ID ........................... 80861229
Category ..................... NET
Board slot type .............. PCI
Readable ID .................. No
Skirt ........................ No
 PCI Function 1 ................................ Enabled


                    Board Information
PCI SCSI Controller
        SCSI A
Manufacturer ................. PCI
ID ........................... 90048078
Category ..................... MSD
Board slot type .............. PCI
Readable ID .................. No
Skirt ........................ No
 PCI Function 1 ................................ Enabled


                    Board Information
Device Configuration for Embedded SCSI
        SCSI A
Manufacturer ................. Hewlett-Packard Co.
ID ........................... ADP7880
Category ..................... MSD
Board slot type .............. Embedded
Readable ID .................. No
Overlay name ................. ADP7880.OVL
Overlay version .............. 1.02
 Wide SCSI Channel Configuration
```

```
   SCSI Channel Interface ..... Wide Channel, Single-
Ended
                                SCSI
   Host Adapter SCSI ID ..... 7
   SCSI Bus Parity Check ..... Enabled
 BIOS configuration ..... Press <Enter> to
configure
 Device configuration ..... Press <Enter> to
configure
 Boot Device option ..... Press <Enter> to
configure
 Utilities ..... Press <Enter> to access


                    Board Information
 PCI SCSI Controller
         SCSI B
 Manufacturer ..... PCI
 ID ..... 90048078
 Category ..... MSD
 Board slot type ..... PCI
 Readable ID ..... No
 Skirt ..... No
 PCI Function 1 ..... Enabled


                    Board Information
 Device Configuration for Embedded SCSI
         SCSI B
 Manufacturer ..... Hewlett-Packard Co.
 ID ..... ADP7880
 Category ..... MSD
 Board slot type ..... Embedded
 Readable ID ..... No
 Overlay name ..... ADP7880.OVL
 Overlay version ..... 1.02
 Wide SCSI Channel Configuration
   SCSI Channel Interface ..... Wide Channel, Single-
Ended
                                SCSI
   Host Adapter SCSI ID ..... 7
   SCSI Bus Parity Check ..... Enabled
 BIOS configuration ..... Press <Enter> to
configure
 Device configuration ..... Press <Enter> to
configure
 Boot Device option ..... Press <Enter> to
configure
 Utilities ..... Press <Enter> to access


                    Board Information
Operating System Optimization
         NOS
Manufacturer ..... Hewlett-Packard Co.
ID ..... HWP21D1
Category ..... OSE
Board slot type ..... Other
Readable ID ..... No
Skirt ..... No
Length ..... 330 millimeters
Overlay name ..... HWP21D1.OVL
Overlay version ..... 0.01
System Optimization ..... Windows NT


                    Board Information
PCI board
         Embedded
Manufacturer ..... PCI
ID ..... 80860008
Category ..... OTH
Board slot type ..... PCI
Readable ID ..... No
Skirt ..... No
PCI Function 1 ..... Enabled


                    Used Resources
 Resource              Slot       Function
 IRQ  0..... System     Reserve Timer ports
 IRQ  1..... System    Reserve Keyboard Controller port
 IRQ  3..... SCSI B     PCI Function 1
 IRQ  4..... SCSI A     PCI Function 1
 IRQ  5..... PCI 5      PCI Function 1
 IRQ  6..... System     On-Board Floppy Controller
 IRQ  8..... System     Reserve Real Time Clock ports
 IRQ  2(9)..... PCI 4      PCI Function 1
 IRQ 10..... PCI 3      PCI Function 1
 IRQ 11..... PCI 2      PCI Function 1
 IRQ 12..... System     Keyboard and Mouse Options
 IRQ 13..... System    Reserve Math Co-processor ports
 IRQ 14..... PCI 6      PCI Function 1
 IRQ 15..... PCI 1      PCI Function 1
```

```
   DMA  2.................. System      On-Board Floppy
Controller
   DMA  4.................. System      Reserve EISA DMA
ports


   Port 0h - 0Fh........... System      Reserve EISA DMA
ports
   Port 20h - 21h.......... System      Reserve ISA PIC
ports
  Port 40h - 43h.......... System    Reserve Timer ports
   Port 60h................ System      Reserve Keyboard
Controller port
   Port 61h................ System      Reserve Speaker
ports
   Port 64h................ System      Reserve Keyboard
Controller port
   Port 70h - 71h.......... System      Reserve Real Time
Clock ports
   Port 80h - 90h.......... System      Reserve EISA DMA
ports
   Port 94h - 9Fh.......... System      Reserve EISA DMA
ports
   Port 0A0h - 0A1h........ System      Reserve ISA PIC
ports
   Port 0C0h - 0DEh........ System      Reserve EISA DMA
ports
   Port 0F0h - 0FFh........ System      Reserve Math Co-
processor ports
   Port 3B4h - 3B5h........ System      Reserve VGA
Resources
   Port 3BAh............... System      Reserve VGA
Resources
   Port 3C0h - 3CAh........ System      Reserve VGA
Resources
   Port 3CCh............... System      Reserve VGA
Resources
   Port 3CEh - 3CFh........ System      Reserve VGA
Resources
   Port 3D4h - 3D5h........ System      Reserve VGA
Resources
   Port 3DAh............... System      Reserve VGA
Resources
   Port 3F0h - 3F5h........ System      On-Board Floppy
Controller
   Port 3F7h............... System      On-Board Floppy
Controller
   Port 40Bh............... System      Reserve EISA DMA
ports
   Port 410h - 43Fh........ System      Reserve EISA DMA
ports
   Port 481h - 483h........ System      Reserve EISA DMA
ports
   Port 487h............... System      Reserve EISA DMA
ports
   Port 489h - 48Ch........ System      Reserve EISA DMA
ports
   Port 4D6h............... System      Reserve EISA DMA
ports
   Port 0C80h - 0C83h...... System      Reserved EISA ID
Ports
  Port 0CA0h - 0CA1h...... System    Reserve IC2 ports
   Port 0CA2h - 0CAFh...... System      Reserved System
Ports
  Port 0CB0h - 0CB7h...... System    Reserved LCD ports


   Port 1C80h - 1C83h...... System      Reserved EISA ID
Ports
   Port 2C80h - 2C83h...... System      Reserved EISA ID
Ports
   Port 3C80h - 3C83h...... System      Reserved EISA ID
Ports
   Port 4C80h - 4C83h...... System      Reserved EISA ID
Ports
  Port 0E0E0h - 0E0FFh.... PCI 6     PCI Function 1
  Port 0E400h - 0E4FFh.... SCSI B    PCI Function 1
  Port 0E800h - 0E8FFh.... SCSI A    PCI Function 1
  Port 0EC00h - 0EC7Fh.... PCI 5     PCI Function 1
  Port 0EC80h - 0ECFFh.... PCI 4     PCI Function 1
  Port 0F880h - 0F8FFh.... PCI 3     PCI Function 1
  Port 0FC00h - 0FC7Fh.... PCI 2     PCI Function 1
  Port 0FC80h - 0FCFFh.... PCI 1     PCI Function 1


   Memory
   Address      Amount
# 0..............640K.... System   Base Memory Options
   0A0000h..........64K.... System      Reserve VGA
Resources
   0B0000h..........64K.... System      Reserve VGA
Resources
```

```
# 0C0000h..........32K.... System     Reserve VGA Resources
# 0E8000h..........96K.... System     Reserve SYSTEM Memory
# 1M..............15M.... System     System Memory 1M - 16M
# 16M.............64M.... System     System Memory 16M - 592M
# 80M.............64M.... System     System Memory 16M - 592M
# 144M............64M.... System     System Memory 16M - 592M
# 208M............64M.... System     System Memory 16M - 592M
# 272M............64M.... System     System Memory 16M - 592M
# 336M............64M.... System     System Memory 16M - 592M
# 400M............64M.... System     System Memory 16M - 592M
# 464M............64M.... System     System Memory 16M - 592M
# 528M............64M.... System     System Memory 16M - 592M
# 592M............64M.... System     System Memory 592M - 1168M
# 656M............64M.... System     System Memory 592M - 1168M
# 720M............64M.... System     System Memory 592M - 1168M
# 784M............64M.... System     System Memory 592M - 1168M
# 848M............64M.... System     System Memory 592M - 1168M
# 912M............64M.... System     System Memory 592M - 1168M
# 976M............64M.... System     System Memory 592M - 1168M
# 1040M...........64M.... System     System Memory 592M - 1168M
# 1104M...........64M.... System     System Memory 592M - 1168M
# 1168M...........64M.... System     System Memory 1168M - 1744M
# 1232M...........64M.... System     System Memory 1168M - 1744M
# 1296M...........64M.... System     System Memory 1168M - 1744M
# 1360M...........64M.... System     System Memory 1168M - 1744M
# 1424M...........64M.... System     System Memory 1168M - 1744M
# 1488M...........64M.... System     System Memory 1168M - 1744M
# 1552M...........64M.... System     System Memory 1168M - 1744M
# 1616M...........64M.... System     System Memory 1168M - 1744M
# 1680M...........64M.... System     System Memory 1168M - 1744M
# 1744M...........64M.... System     System Memory 1744M - 2320M
# 1808M...........64M.... System     System Memory 1744M - 2320M
# 1872M...........64M.... System     System Memory 1744M - 2320M
# 1936M...........64M.... System     System Memory 1744M - 2320M
# 2000M...........64M.... System     System Memory 1744M - 2320M
# 2064M...........64M.... System     System Memory 1744M - 2320M
# 2128M...........64M.... System     System Memory 1744M - 2320M

# 2192M...........64M.... System     System Memory 1744M - 2320M
# 2256M...........64M.... System     System Memory 1744M - 2320M
# 2320M...........64M.... System     System Memory 2320M - 2896M
# 2384M...........64M.... System     System Memory 2320M - 2896M
# 2448M...........64M.... System     System Memory 2320M - 2896M
# 2512M...........64M.... System     System Memory 2320M - 2896M
# 2576M...........64M.... System     System Memory 2320M - 2896M
# 2640M...........64M.... System     System Memory 2320M - 2896M
# 2704M...........64M.... System     System Memory 2320M - 2896M
# 2768M...........64M.... System     System Memory 2320M - 2896M
# 2832M...........64M.... System     System Memory 2320M - 2896M
# 2896M...........64M.... System     System Memory 2896M - 3472M
# 2960M...........64M.... System     System Memory 2896M - 3472M
# 3024M...........64M.... System     System Memory 2896M - 3472M
# 3088M...........64M.... System     System Memory 2896M - 3472M
# 3152M...........64M.... System     System Memory 2896M - 3472M
# 3216M...........64M.... System     System Memory 2896M - 3472M
# 3280M...........64M.... System     System Memory 2896M - 3472M
# 3344M...........64M.... System     System Memory 2896M - 3472M
# 3408M...........64M.... System     System Memory 2896M - 3472M
# 3472M...........64M.... System     System Memory 3472M - 4048M
# 3536M...........64M.... System     System Memory 3472M - 4048M
# 3600M...........64M.... System     System Memory 3472M - 4048M
# 3664M...........64M.... System     System Memory 3472M - 4048M
# 3728M...........64M.... System     System Memory 3472M - 4048M
# 3792M...........64M.... System     System Memory 3472M - 4048M
# 3856M...........64M.... System     System Memory 3472M - 4048M
# 3920M...........64M.... System     System Memory 3472M - 4048M
# 3984M...........64M.... System     System Memory 3472M - 4048M
# 4048M...........16M.... System     System Memory 4048M - 4624M
  0FE0FF000h........4K.... PCI 6      PCI Function 1
  4071M.............1M.... PCI 6      PCI Function 1
  0FE8FA000h........4K.... SCSI B     PCI Function 1
  0FE8FB000h........4K.... SCSI A     PCI Function 1
  0FEC00400h........3K.... System     Reserve P6 resources
  0FEC01000h........1K.... Embedded   PCI Function 1
  0FEC01400h....19963K.... System     Reserve P6 resources
  0FFF80000h......512K.... System     Reserve SYSTEM Memory

# = Caching
```

Available Resources			
IRQs	DMAs	ISA I/O Ports	Memory Amount	Address
7	0	10h - 1Fh	32K	0C8000h
	1	22h - 3Fh	64K	0D0000h
	3	44h - 5Fh	32K	0E0000h
	5	62h - 63h		
	6	65h - 6Fh		
	7	72h - 7Fh		
		91h - 93h		
		0A2h - 0BFh		
		0DFh - 0EFh		
		100h - 3B3h		
		3B6h - 3B9h		
		3BBh - 3BFh		
		3CBh		

```
|       |    3CDh          |
|       |    3D0h -   3D3h  |
|       |    3D6h -   3D9h  |
|       |    3DBh -   3EFh  |
|       |    3F6h          |
|       |    3F8h -   400h  |
```

System Specifications

Slot Name	Slot Type	Board ID	Accept Skirted	Max Length	Bus-master	Slot Tag(s)
PCI 1	PCI	101e9010	Yes	341mm	Yes	
PCI 2	PCI	101e9010	Yes	341mm	Yes	
PCI 3	PCI	101e9010	Yes	341mm	Yes	
PCI 4	PCI	101e9010	Yes	341mm	Yes	
PCI 5	PCI	101e9010	Yes	341mm	Yes	
PCI 6	PCI	80861229	Yes	341mm	Yes	
EISA 1	EISA	(Empty)	Yes	341mm	Yes	
EISA 2	EISA	(Empty)	Yes	341mm	Yes	
EISA 3	EISA	(Empty)	Yes	341mm	Yes	
EISA 4	EISA	(Empty)	Yes	341mm	Yes	
SCSI A	PCI	90048078	Yes	341mm	Yes	
SCSI A	Other	ADP7880	Yes	341mm	No	ADP7880
SCSI B	PCI	90048078	Yes	341mm	Yes	
SCSI B	Other	ADP7880	Yes	341mm	No	ADP7880
NOS	Other	HWP21D1	Yes	341mm	No	HWP21D1

Nonvolatile memory 8K

Disk Array Configuration Parameters

```
    MegaRAID DiskArray Display Utility version 1.01
    HA #2 FwVersion = Xi75, BIOS Version = 1.42, DRAM Size
= 16MB
    HA #3 FwVersion = Xi75, BIOS Version = 1.42, DRAM Size
= 16MB
    HA #4 FwVersion = Xi75, BIOS Version = 1.42, DRAM Size
= 16MB
    HA #5 FwVersion = Xi75, BIOS Version = 1.42, DRAM Size
= 16MB
    HA #6 FwVersion = Xi75, BIOS Version = 1.42, DRAM Size
= 16MB
    No of RAIDCard Adapters Found = 5
    ADAPTER MAPPINGS :
    Adapter = 0  Device Location = \\.\Scsi2:
    Adapter = 1  Device Location = \\.\Scsi3:
    Adapter = 2  Device Location = \\.\Scsi4:
    Adapter = 3  Device Location = \\.\Scsi5:
    Adapter = 4  Device Location = \\.\Scsi6:
------------------------------------------------------------
-----
     MEGARAID HA-0
    LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0Spans : 3  Read Ahead : No
Stripe Size : 8K Status : Optimal Write Mode : Write Through
DirectIO : Enabled Num Stripes : 6
     Physical Drives ### Channel  ID    ConfiguredSize
                              0       0       8678 MB
                              0       1       8678 MB
                              0       2       8678 MB
                              0       3       8678 MB
                              0       4       8678 MB
                              0       6       8678 MB
                              1       0       8678 MB
                              1       1       8678 MB
                              1       2       8678 MB
                              1       3       8678 MB
                              1       4       8678 MB
                              1       6       8678 MB
```

```
                        2       0       8678 MB
                        2       1       8678 MB
                        2       2       8678 MB
                        2       3       8678 MB
                        2       4       8678 MB
                        2       6       8678 MB
------------------------------------------------------------------
     MEGARAID HA-1
    LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0Spans : 3  Read Ahead : No
Stripe Size : 8K Status : Optimal Write Mode : Write Through
DirectIO : Enabled Num Stripes : 6
     Physical Drives ### Channel  ID    ConfiguredSize
                        0       1       8678 MB
                        0       2       8678 MB
                        0       3       8678 MB
                        0       4       8678 MB
                        0       5       8678 MB
                        0       6       8678 MB
                        1       1       8678 MB
                        1       2       8678 MB
                        1       3       8678 MB
                        1       4       8678 MB
                        1       5       8678 MB
                        1       6       8678 MB
                        2       0       8678 MB
                        2       1       8678 MB
                        2       2       8678 MB
                        2       3       8678 MB
                        2       4       8678 MB
                        2       6       8678 MB
------------------------------------------------------------------
     MEGARAID HA-2
    LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0Spans : 3  Read Ahead : No
Stripe Size : 8K Status : Optimal Write Mode : Write Through
DirectIO : Enabled Num Stripes : 6
     Physical Drives ### Channel  ID    ConfiguredSize
                        0       0       8678 MB
                        0       1       8678 MB
                        0       2       8678 MB
                        0       3       8678 MB
                        0       4       8678 MB
                        0       6       8678 MB
                        1       0       8678 MB
                        1       1       8678 MB
                        1       2       8678 MB
                        1       3       8678 MB
                        1       4       8678 MB
                        1       6       8678 MB
                        2       0       8678 MB
                        2       1       8678 MB
                        2       2       8678 MB
                        2       3       8678 MB
                        2       4       8678 MB
                        2       6       8678 MB
------------------------------------------------------------------
     MEGARAID HA-3
    LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0Spans : 3  Read Ahead : No
Stripe Size : 8K Status : Optimal Write Mode : Write Through
DirectIO : Enabled Num Stripes : 6
     Physical Drives ### Channel  ID    ConfiguredSize
                        0       1       4067 MB
                        0       2       4067 MB
                        0       3       4067 MB
                        0       4       4067 MB
                        0       5       4067 MB
                        0       6       4067 MB
                        1       1       4067 MB
                        1       2       4067 MB
                        1       3       4067 MB
                        1       4       4067 MB
                        1       5       4067 MB
                        1       6       4067 MB
                        2       1       4067 MB
                        2       2       4067 MB
                        2       3       4067 MB
                        2       4       4067 MB
                        2       5       4067 MB
                        2       6       4067 MB
------------------------------------------------------------------
     MEGARAID HA-4
    LogicalDrives Found  = 1
Logical Drive : 1
Raid Level : 0Spans : 3  Read Ahead : No
Stripe Size : 8K Status : Optimal Write Mode : Write Through
DirectIO : Enabled Num Stripes : 6
     Physical Drives ### Channel  ID    ConfiguredSize
                        0       1       4067 MB
                        0       2       4067 MB
```

```
0       3       4067 MB
0       4       4067 MB
0       5       4067 MB
0       6       4067 MB
1       1       4067 MB
1       2       4067 MB
1       3       4067 MB
1       4       4067 MB
1       5       4067 MB
1       6       4067 MB
2       1       4067 MB
2       2       4067 MB
2       3       4067 MB
2       4       4067 MB
2       5       4067 MB
2       6       4067 MB
```

Tuxedo UBBconfig

```
# configuration file for client1
*RESOURCES
IPCKEY123456
DOMAINIDtpcc
MASTERtpcc
MAXACCESSERS512
MAXSERVERS128
MAXSERVICES512
MODELSHM
LDBALN
*MACHINES
DEFAULT:
    APPDIR="c:\temp\tux"
    TUXCONFIG="c:\temp\tux\tuxconfig"
    TUXDIR="c:\tuxedo"
NRDWBC1 LMID=tpcc
*GROUPS
GROUP1
    LMID=tpccGRPNO=1OPENINFO=NONE
*SERVERS
DEFAULT:
    CLOPT="-A"
tux_serverSRVGRP=GROUP1 SRVID=1 RQADDR=tpcc REPLYQ=Y MIN=37
*SERVICES
NEW_ORDER
STOCK_LEVEL
PAYMENT
ORDER_STATUS
```

HP NetServer E40 Configurations - Clients

```
Microsoft Diagnostics Report For \\NRDWBC1
------------------------------------------------------------
----------

OS Version Report
------------------------------------------------------------
----------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381) x86 Uniprocessor Free
Registered Owner: NSD Performance Lab, Hewlett Packard
Product Number: 50370-419-0240645-79481
------------------------------------------------------------
----------

System Report
------------------------------------------------------------
----------
System: AT/AT COMPATIBLE
```

```
Hardware Abstraction Layer: PC Compatible Eisa/Isa HAL
BIOS Date: 01/27/97
BIOS Version: PhoenixBIOS Version 4.05.8
              Ph

Processor list:
   0:  x86 Family 6 Model 1 Stepping 9 GenuineIntel ~199 Mhz
------------------------------------------------------------------------

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

Adapter:
   Setting: 1152 x 864 x 256
            70 Hz
   Type: cirrus compatible display adapter
   String: Cirrus Logic Compatible
   Memory: 1 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 - FAT)  Total: 2,080,096KB, Free: 854,944KB
E:  (Remote - NTFS) \\hpdpc338\perfdata  Total: 12,493,628KB, Free:
2,492,316KB
U:  (Remote - NTFS) \\hpdpc338\perfdata  Total: 12,493,628KB, Free:
2,492,316KB

Memory Report
------------------------------------------------------------------------
Handles: 19,828
Threads: 287
Processes: 116

Physical Memory (K)
   Total: 261,556
   Available: 133,748
   File Cache: 12,232
```

```
Services Report
------------------------------------------------------------------------
Ataman TCP Remote Logon Services              Running   (Automatic)
EventLog (Event log)                          Running   (Automatic)
Gopher Publishing Service                     Running   (Automatic)
Server                                        Running   (Automatic)
Workstation (NetworkProvider)                 Running   (Automatic)
FTP Publishing Service                        Running   (Automatic)
NT LM Security Support Provider               Running   (Manual)
Remote Command Service                        Running   (Automatic)
Remote Procedure Call (RPC) Service           Running   (Automatic)
TUXEDO IPC HELPER                             Running   (Automatic)
World Wide Web Publishing Service             Running   (Automatic)


Drivers Report
------------------------------------------------------------------------
AFD Networking Support Environment (TDI)      Running   (Automatic)
aic78xx (SCSI miniport)                       Running   (Boot)
atapi (SCSI miniport)                         Running   (Boot)
Beep (Base)                                   Running   (System)
Cdfs (File system)                            Running   (Disabled)
Cdrom (SCSI CDROM Class)                      Running   (System)
cirrus (Video)                                Running   (System)
Disk (SCSI Class)                             Running   (Boot)
Fastfat (Boot file system)                    Running   (Disabled)
Floppy (Primary disk)                         Running   (System)
HP 10/100TX PCI LAN Adapter Driver (NDIS)     Running   (Automatic)
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running
(System)
Keyboard Class Driver (Keyboard Class)        Running   (System)
KSecDD (Base)                                 Running   (System)
Mouse Class Driver (Pointer Class)            Running   (System)
Msfs (File system)                            Running   (System)
Mup (Network)                                 Running   (Manual)
Microsoft NDIS System Driver (NDIS)           Running   (System)
NetBIOS Interface (NetBIOSGroup)              Running   (Manual)
WINS Client(TCP/IP) (PNP_TDI)                 Running   (Automatic)
Npfs (File system)                            Running   (System)
Null (Base)                                   Running   (System)
Parallel (Extended base)                      Running   (Automatic)
Parport (Parallel arbitrator)                 Running   (Automatic)
ParVdm (Extended base)                        Running   (Automatic)
Rdr (Network)                                 Running   (Manual)
Serial (Extended base)                        Running   (Automatic)
```

```
Srv (Network)                                   Running
(Manual)
TCP/IP Service (PNP_TDI)                        Running
(Automatic)


IRQ and Port Report
-------------------------------------------------------------
----------
Devices                         Vector Level  Affinity
-------------------------------------------------------------
----------
i8042prt                             1     1 0xffffffff
i8042prt                            12    12 0xffffffff
Serial                               4     4 0x00000000
Serial                               3     3 0x00000000
Floppy                               6     6 0x00000000
HPTX                                 9     9 0x00000000
HPTX                                10    10 0x00000000
aic78xx                             11    11 0x00000000
atapi                                0    15 0x00000000
-------------------------------------------------------------
----------
Devices                         Physical Address  Length
-------------------------------------------------------------
----------
i8042prt                          0x00000060  0x0000000001
i8042prt                          0x00000064  0x0000000001
Parport                           0x00000378  0x0000000003
Serial                            0x000003f8  0x0000000007
Serial                            0x000002f8  0x0000000007
Floppy                            0x000003f0  0x0000000006
Floppy                            0x000003f7  0x0000000001
HPTX                              0x0000f8e0  0x0000000014
HPTX                              0x0000f8c0  0x0000000014
aic78xx                           0x0000fc00  0x0000000100
atapi                             0x00000170  0x0000000008
cirrus                            0x000003b0  0x000000000c
cirrus                            0x000003c0  0x0000000020


DMA and Memory Report

-------------------------------------------------------------
----------
Devices                         Channel    Port
-------------------------------------------------------------
----------
Floppy                             2     0
-------------------------------------------------------------
----------
Devices                         Physical Address  Length
-------------------------------------------------------------
----------
HPTX                            0xfecfe000  0x00000014
HPTX                            0xfecfe000  0x00000014
HPTX                            0xfecfd000  0x00000014
HPTX                            0xfecfd000  0x00000014
aic78xx                         0xfecff000  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;C:\WINNT\system32;C:\WINNT;;C:\MSSQL\BINN;C:\T
UXEDO\bin;c:\PROGRA~1\DEVSTU~1\SHARED~1\bin\ide;c:\PROGRA~1\
DEVSTU~1\SHARED~1\bin;c:\PROGRA~1\DEVSTU~1\vc\bin;;C:\NTRESK
IT
   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=1
   ROOTDIR=C:/
   SHELL=C:/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\sharedide\bin\ide;c:\program
files\devstudio\sharedide\bin;c:\program files\devstudio\vc\bin
   lib=c:\program files\devstudio\vc\lib;c:\program
files\devstudio\vc\mfc\lib;c:\program files\devstudio\vc\lib;c:\program
files\devstudio\vc\mfc\lib;%lib%
   include=c:\program files\devstudio\vc\include;c:\program
files\devstudio\vc\atl\include;c:\program
files\devstudio\vc\mfc\include;c:\program
files\devstudio\vc\include;c:\program
files\devstudio\vc\atl\include;c:\program
files\devstudio\vc\mfc\include;%include%


Network Report
------------------------------------------------------------------------
Your Access Level: Admin & Local
Workgroup or Domain: WORKGROUP
Network Version: 4.0
LanRoot: WORKGROUP
Logged On Users: 6
Current User (1): Administrator
  Logon Domain: NRDWBC1
  Logon Server: NRDWBC1
Current User (2): IUSR_NRDWBC1
  Logon Domain: NRDWBC1
  Logon Server: NRDWBC1
Current User (3): Administrator
  Logon Domain: NRDWBC1
  Logon Server: NRDWBC1
Current User (4): tpcc
  Logon Domain: NRDWBC1
  Logon Server: NRDWBC1
Current User (5): tpcc
  Logon Domain: NRDWBC1
  Logon Server: NRDWBC1
Current User (6): IUSR_NRDWBC1
  Logon Domain: NRDWBC1
  Logon Server: NRDWBC1
Transport: NetBT_HPTX1, 00-60-B0-3C-41-B3, VC's: 1, Wan: Wan
Transport: NetBT_HPTX2, 00-60-B0-3C-A5-1C, 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
Bytes Received: 952,870
SMB's Received: 9,210
Paged Read Bytes Requested: 40,960
Non Paged Read Bytes Requested: 34,888
Cache Read Bytes Requested: 15,603
Network Read Bytes Requested: 48,745
Bytes Transmitted: 947,513
SMB's Transmitted: 9,194
Server File Opens: 13
```

Appendix D – Disk Storage

180-day and 8 hour Space Calculations are provided below:

Note: Numbers are in Kbytes unless otherwise specified					
180 day growth					
Warehouses	820	tpmC	9228.7		
Table	Rows	Data	Index	Extra 5%	Data + Index + 5%
warehouse	820	1640	10	83	1733
district	8200	16400	70	824	17294
customer	24600000	16403280	98822	825105	17327207
history	24600000	1230002	0		1230002
new_order	7380000	82000	500	4125	86625
orders	24600000	639600	3858		643458
order_line	246004079	13674890	89384		13764274
item	100000	9100	46	457	9603
stock	82000000	27338800	151050	1374493	28864343
Totals		59395712	343740	2205086	61944538
Segment			Size		
master & msdb & model & tempdb			60000		
misc_seg			8250000		
ordln_seg			23400000		
cs_seg			49100000		
Total			80810000		
Dynamic Space	15544492	Sum of Data for order, order_line and history			
Static Space	46400046	Sum of all data and index +5 % - dynamic space			
Free Space	18865462	Total space allocated to DBMS - dynamic - static			
Daily Growth	2799131	Dynamic Space*tpmC/(W*62.5)			
Daily Spread	14666766	Free Space - 1.5 * Daily Growth			
		This can be reconfigured to eliminate daily spread (zero assumed)			
180 day space	550243589.3	Static space + 180*(daily growth + daily spread)			
8 hr log space	24104314				
Disk Allocation	Space Needed	Disk Size		Disks Needed	
180 day space	550243589	4160512	RAID 0	17	
		8885248	RAID 0	54	
8 hr log space	24104314	8885248	RAID 1	6	
OS and swap	2400000	4160512	RAID 0	1	

All numbers are in kbytes unless otherwise noted						
8 hour space						
Warehouses	820	tpmC	9198.37			
Table	Space	Space	Space	# of New	Pages per	8 Hour
	Before	After	Used	Order Trns	New Order Trn	Space
Syslogs	5665974	8974940	3308966	608106	5.441429619	24025096
History	1279988	1309142	29154	608106	0.0479423	211676
Orders	697378	705702	8324	608106	0.013688403	60437
Order_line	14353772	14691862	338090	608106	0.55597215	2454738

Appendix E – Quotations

All quotes can be found on the following pages.

Software House International **Pricing Proposal**	**Quotation #MO-970212-35960**

SHI Account Exec: Matthew O. Martin

HEWLETT-PACKARD
NETWORK SERVER DIVISION
LARRY GRAY
August 18th, 1997 LX9198
Prices good for sixty days.
Phone: 408-553-4186
Fax: 408-553-2902

Telephone : (408) 232-0850
Fax : (408) 526-1222

Reference:

Description	Part #	Qty	List$	Customer$	Extended
HP NetServer LX Pro 6/200	D4311B	1	$0.00	$11,305.00	$11,305.00
NetServer LX 6/200 Dual Processor Card	D4866A	1	$0.00	$825.00	$825.00
NetServer LX 6/200 Pentium Pro Chip	D4867A	3	$0.00	$2,145.00	$6,435.00
4GB Memory Board	D4967A	1	$0.00	$8,025.00	$8,025.00
512 MByte ECC DIMM	D4968A	7	$0.00	$8,350.00	$58,450.00
Internal SCSI Cable	5182-6740	4	$0.00	$30.00	$120.00
External SCSI Cable	5182-6737	17	$0.00	$55.00	$935.00
HP Storage System 6	D3604A	17	$0.00	$995.00	$16,915.00
HP 9GB SCSI-2 Hot Swap	D4289A	40	$0.00	$2,045.00	$81,800.00
HP 4.2 GB SCSI-2 Hot Swap	D3583B	66	$0.00	$1,140.00	$75,240.00
HP SureStore Tape 6000i	C1528F	1	$0.00	$905.00	$905.00
24 Port 10Base-T EtherHub	347465	362	$0.00	$179.00	$64,798.00
HP NetServer E40	D4933A	4	$0.00	$1,865.00	$7,460.00
64MB RAM SIMMS	D4290A	22	$0.00	$865.00	$19,030.00
HP 10/100 PCI Network Adaptor	J3171A	13	$0.00	$72.00	$936.00
HP 14" VGA Monitor	D2813S	9	$0.00	$245.00	$2,205.00
SQL Server Programmers Toolkit	Z50002	1	$0.00	$129.00	$129.00
Visual C++ v.4.0	Z50003	1	$0.00	$425.00	$425.00
AMI Megaraid Ultra GT F/W SCSI Controller	432110000	5	$0.00	$1,795.00	$8,975.00
Windows NT 4.0	Z50001	4	$0.00	$587.00	$2,348.00

Software House International Pricing Proposal	Quotation #MO-970212-35960

HEWLETT-PACKARD
NETWORK SERVER DIVISION
LARRY GRAY
August 18th, 1997 LX9198
Prices good for sixty days.
Phone: 408-553-4186
Fax: 408-553-2902

SHI Account Exec: Matthew O. Martin

Telephone : (408) 232-0850
Fax : (408) 526-1222

Reference:

Description	Part #	Qty	List$	Customer$	Extended
SQL Server w/ unlimited clients	810-00043	1	$0.00	$24,300.00	$24,300.00

Total $391561.00

Additional Comments:

Redmond, WA 98052-6399 Fax 206 936 7329

Microsoft

June 25, 1997

Mr. Larry Gray
Project Manager
Hewlett-Packard Company
Network Server Division
5301 Stevens Creek Blvd.
Santa Clara, CA 85052

via FAX # 408-553-2902

Dear Larry,

Microsoft has received your request for permission to disclose results of TPC-C benchmarks done with the following HP system and Microsoft SQL Srever 6.5, Enterprise Edition:

HP NetServer LX Pro 6/200, 4-processor, Pentium Pro-based, 200 MHz
Performance: over 9000 tpmC, price/performance: $50/tpmC approx.

Microsoft hereby grants HP permission to disclose these results and acknowledges that HP has formally requested permission to do so in accordance with the license agreement for Microsoft SQL Server software.

Best Regards,

Sid Arora
Product Manager, Microsoft SQL Server
Personal and Business Systems Group

Microsoft®

August 12, 1997

Mr. Larry Gray
Product Manager
Hewlett-Packard Company
Network Server Division
5301 Stevens Creek Blvd.
Santa Clara, CA 95052

via FAX # 408-553-2902

Dear Larry,

Here is the information you requested regarding pricing of certain Microsoft products:

Microsoft SQL Server 6.5, Enterprise Edition, unlimited user licence	$28999
Microsoft Windows NT Server 4.0, Enterprise Edition, incl 25 CALs	$3999
Windows NT Server 4.0 software, incl 5 CALs	$809
Microsoft SQL Workstation (includes programmers toolkit)	$499
Visual C++ 32-bit edition (subscription)	$499
5-yr maintenance for above software @ $2095/yr	$10475

This quote is valid for the next 60 days. Please let me know if I can be of any further assistance.

Best regards,

Sid Arora
Product Manager, Microsoft SQL Server
Personal and Business Systems Group

BEA PRICE QUOTE:

6/10/97
For: Mr. Larry Gray, HP
By: Jeff Hartson, BEA

TUX 6.3 LIC DESC	# of Machines	Machine Class	Users		List Price		5 year maintenance
TUXEDO DEVELOPMENT			1	$	2,395.00	$	1,796.25
TUXEDO RUNTIME	4	Tier 1 Machine		$	12,000.00	$	9,000.00
			Total	$	14,395.00	$	10,796.25