HEWLETT® PACKARD

TPC Benchmark® C
Full Disclosure Report

HP NetServer LXr 8000

using Microsoft SQLServer 7.0 Enterprise Edition
on Microsoft NT 4.0 Enterprise Edition

Third Edition
January 18, 1999

Third Edition (*) - January 18, 1999
First Printing.

Printed in U.S.A., January 18, 1999

(*) Explanatory statement about the third edition

This is an FDR correction from the January 5, 1999 TPC-C results from Hewlett-Packard. The second edition of the FDR was found to be in insignificant deviation from the TPC-C specification 3.4, specifically clause 8.1.1.6, for not including the boot.ini file in the FDR. (See TAB ID 278 motion #20 and TAB ID 279B motion #23 in the minutes of the 57th TPC General Meeting).

Action was taken by Hewlett-Packard to include the boot.ini file in this edition.

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark® C test conducted on the HP NetServer LXr 8000 in a client/server configuration, using Microsoft SQLServer 7.0 Enterprise Edition and TUXEDO Transaction Monitor. The operating system used for the benchmark was Microsoft NT 4.0 Enterprise Edition. The application was written in C and compiled using Microsoft Visual C++.

TPC Benchmark® C Metrics

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

Standard and Executive Summary Statements

The following pages contain the executive summary of the benchmark results for the HP NetServer LXr 8000 system. The Standard System Summary is given below.

Company Name	System Name	Database Software	Operating System
Hewlett-Packard Company	HP NetServer LXr 8000	Microsoft SQLServer 7.0 Enterprise Edition	Microsoft NT 4.0 Enterprise Edition
Total System Cost	**TPC-C Throughput**	**Price/Performance**	**Availability Date**
$621,784	23143 tpmC	$26.87 per tpmC	February 1, 1999

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 Tom Sawyer of Performance Metrics, Inc. to verify compliance with the relevant TPC specifications.

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

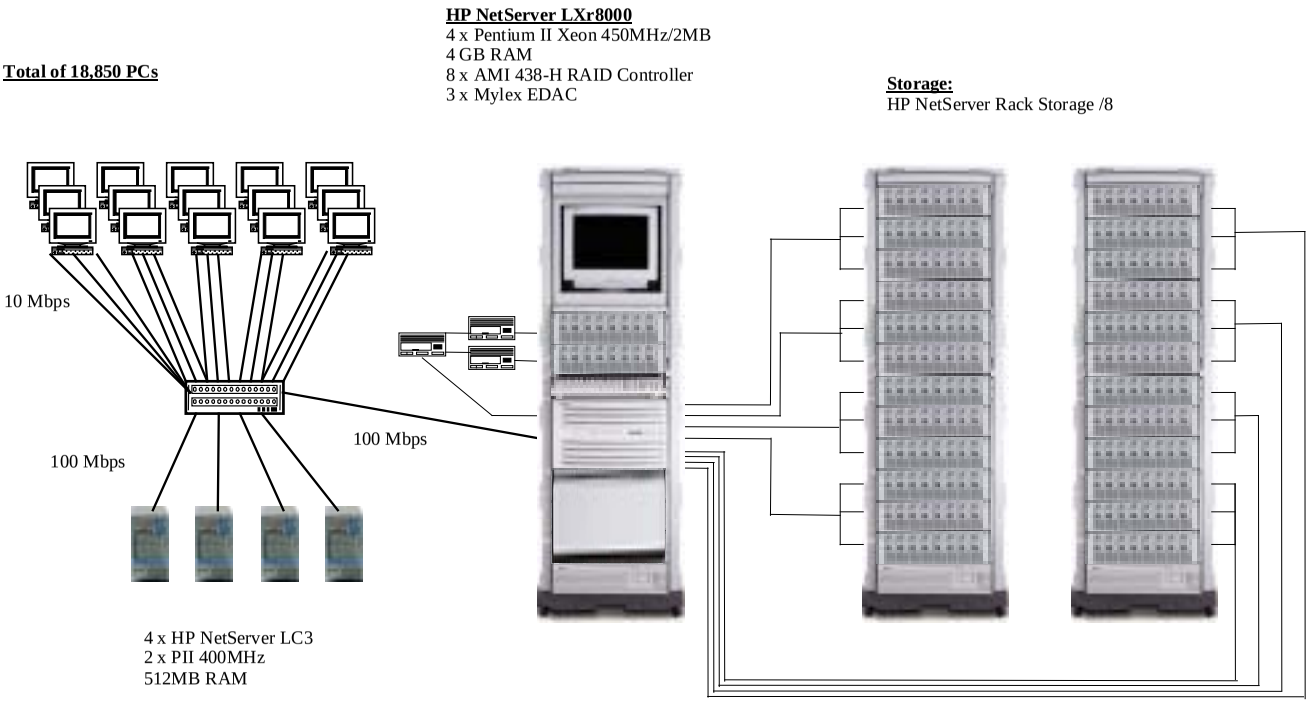
or

Hewlett-Packard Company
Enterprise NetServer Division
10955 Tantau Avenue
Cupertino, CA 95014-0770 USA
Attn:Neil MacDonald, MS 45NUH

HP NetServer LXr 8000 Client/Server

TPC-C Revision 3.4

Total System Cost	TPC Throughput	Price/Performance	Report Date	Availability Date
$621,784	23143 tpmC	$26.87 per tpmC	January 18, 1999	February 1, 1999
Processors	**Database Manager**	**Operating System**	**Other Software**	**Number of Users**
4 Intel Pentium II Xeon 450MHz 2 Mbyte L2	Microsoft SQLServer 7.0 Enterprise Edition	Microsoft NT 4.0 Enterprise Edition	Microsoft Visual C++ TUXEDO Transaction Monitor	18850



System Components	Server		Each Client	
	Qty	Type	Qty	Type
Processors	4	450MHz Intel Pentium II Xeon	2	400MHz Pentium II
Cache Memory	each	2 Mbyte L2 cache	each	512kbyte L2 Cache
Memory	4	Gbyte	512	Mbyte
Disk Controllers	8	AMI Series 438-H Super High Perfo	1	HP SCSI-2 Controller
Disk Drives	184	HP Hot-swap 9Gbyte 10krpm SCSI	1	4 Gbyte disk
	24	HP Hot-swap 4Gbyte 10krpm SCSI		
	1	HP Hot-swap 9Gbyte Ultra2 10krpm SCSI		
Total Storage	1.67	TB		
Tape Drives	1	HP Surestore DAT24i		
Terminals	1	Console Terminal	1	Console Terminal

HEWLETT® PACKARD

HP NetServer LXr 8000

TPC-C Rev 3.4

Report Date: January 18, 1999

Description	Part Number	Brand	Price Key	Unit Price	Qty	Extended Price	Maint. Price
HP NetServer LXr 8000	D6021A	HP	1	11,055	1	11055	
Pentium II Xeon 450MHz 2MB Cache	D6122A	HP	2	8899	4	35596	
1Gbyte DIMM kit	D6114A	HP	1	4280	4	17120	
Rkmt HP Ultra VGA 1280 17" Display	D2818A	HP	1,2	213	1	213	168
HP NetServer mini-DIN keyboard and mouse	D4950B/C3751B	HP	1	79	1	79	
HP 9.1 GB 10K HotSwap Wide Ultra2 SCSI Disk	D6107A	HP	1	1022	1	1022	
HP NetServer 10/100TX PCI LAN Adapter	D5013A	HP	1	82	1	82	
HP Keyboard Rack Mount Kit	E7713A	HP	1	150	1	150	
2.0 Meter Rack w/side panels,rear door+fan+pdu	J1487B/E4471A/E7	HP	1	1460	3	4380	
HP Power Distribution Unit 120-240V	E5929A	HP	1	234	5	1170	
HP 19" Plain Shelf, for racks	E3666A	HP	1	61	1	61	
APC Smart UPS 3000NS 208V 3000VA	588293	APC	1,6	1728	3	5184	1170
HP SureStore DAT24i Internal Tape Drive	C1555C	HP	1	960	1	960	
HP NetServer Lxr 8000 Warranty Upgrade	H5516A	HP	1,2				4286
				Server Hardware Subtotal		77072	5624
HP NetServer Rack Storage/8+spares	D4902A	HP	1	2080	29	60320	
HP NetServer Rack Storage/8 Support	D4902A+02A	HP	2				27840
HP 4.2GB, 10krpm Hot-swap disk module+spares	D4903A	HP	1	657	27	17739	
HP 9GB, 10krpm Hot-swap disk module+spares	D6019A	HP	1	1030	203	209090	
AMI Series 438-H Super High Performance RAID Contr	4383508116	AMI	7	2075	10	20750	620
Mylex DACSXI EDAC 5 chnl 32MB	Z75091	Mylex	1,4	3569	5	17845	995
HP SCSI Cable 2.5m UDHTS 68/HDTS 68	D3637C	HP	1	82	18	1476	
HP SCSI Cable 2.5m HDTS68/HDTS68	D3636C	HP	1	82	13	1066	
HP SCSI Cable 0.9m	C2911A	HP	1	122	4	488	
HP Storage System/6	D3604B	HP	1,2	997	2	1994	3118
				Storage Subtotal		330768	32573
Microsoft SQL Server 7.0 Enterprise Edition		MS	3	28999	1	28999	10475
Microsoft Windows NT Server 4.0 Enterprise Edition	Z75092	MS	1	3330	1	3330	
MKS Toolkit	323795	MKS	1	258	1	258	
				Server Software Subtotal		32587	10475
HP Netserver LC 3	D6126T	HP	1	2455	4	9820	
HP Netserver LC 3 Warranty Upgrade	H5519A	HP	2				6956
LC 3 Pentium II 400-Mhz	D6092A	HP	1	1138	4	4552	
128MB DIMMs for LC 3	D6098A	HP	1	445	16	7120	
HP NetServer 10/100TX PCI LAN Adapter	D5013A	HP	1	82	4	328	
HP Ultra VGA 1280 17" Display	D2818A	HP	1,2	213	4	852	672
				Client Hardware Subtotal		22672	7628
Microsoft Windows NT Server 4.0	Z75093	MS	1	568	4	2272	
Microsoft Visual C++	716856	MS	1	449	1	449	
Tuxedo Run Time		BEA	5	3000	4	12000	9000
				Client Software Subtotal		14721	9000
HP Procurve Switch 2400M	J4122A	HP	1,2	1189	2	2378	1680
LANTech 8-port 10baseT hub (8+1 ports)		Lantech	1	28.75	2595	74606	
				Connectivity Subtotal		76984	1680
					Total	**$554,804**	**$66,980**

Notes: Price key: 1=Software House, 2=Price from HP Corporate Price list
3=Microsoft, 4=Mylex, 5=BEA, 6 = APC, 7 = AMI
Audited by Tom Sawyer Performance Metrics, Inc.

5-yr Cost of Ownership:	**$621,784**
tpmC:	**23143.00**
$/tpmC:	**26.87**

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

Numerical Quantities Summary for HP NetServer LXr 8000

MQTH, Computed Maximum Qualified Throughput **23143.00 tpmC**

Response Times (in seconds)

	90th %-ile	Maximum	Average
New-Order	0.92s	13.35s	0.58s
Payment	0.69s	3.16s	0.39s
Order-Status	0.80s	12.82s	0.48s
Delivery (interactive portion)	0.38s	1.80s	0.22s
Delivery (deferred portion)	0.99s	2.69s	0.65s
Stock-Level	3.61s	8.62s	2.91s
Menu	0.60s	2.63s	0.28s

Response time delay added for emulated components 0.1 seconds

Transaction Mix, in percent of total transactions

New-Order	44.87%
Payment	43.02%
Order-Status	4.03%
Delivery	4.03%
Stock-Level	4.05%

Keying/Think Times

	Keying Time			Think Time		
	Min	Avg	Max	Min	Avg	Max
New-Order	18.02s	18.03s	18.18s	0.01s	12.15s	164.79s
Payment	3.01s	3.02s	3.17s	0.01s	12.17s	176.39s
Order-Status	2.01s	2.02s	2.16s	0.01s	10.18s	117.89s
Delivery (interactive)	2.01s	2.02s	2.16s	0.01s	5.16s	55.92s
Stock-Level	2.01s	2.02s	2.17s	0.01s	5.2s	58.92s

Test Duration

Ramp up time	63.13 minutes
Measurement interval	29 minutes
Transactions during measurement interval	1495660
Ramp down time	37.87 minutes

Checkpointing

Number of checkpoints in measurement interval	1
Checkpoint Interval	29 minutes

Reproducibility Run

Throughput	23038.69 tpmC
Relative to MQTH	-0.45%

Preface

This Report

This is the full disclosure report for a benchmark test of the HP NetServer LXr 8000 using Microsoft SQLServer 7.0 Enterprise Edition. It meets the requirements of the TPC Benchmark ® C Standard Specification, Revision 3.4 dated August 25, 1998. 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 HP NetServer LXr 8000. The full configuration was built by adding additional memory, additional disk adapters and drives, and a network adapter. The operating system used was Microsoft NT 4.0 Enterprise Edition and the database was Microsoft SQLServer 7.0 Enterprise Edition.

The processor architecture of the HP NetServer LXr 8000 was designed for the Intel Pentium II Xeon processor and associated chipset. The HP NetServer LXr 8000 used in this test was powered by 4 450MHz Intel Pentium II Xeon processors, each with 2 Mbyte of 2nd level cache.

This configuration used 4 Gbyte of HP RAM. This was achieved with 16 256 Mbyte DIMMs populating 2 memory boards.

This configuration also used Ultra2 SCSI PCI disk controllers embedded in the motherboard, 8 AMI Series 438-H Super High Performance RAID Controller PCI Disk Array Controllers (DACs), and 3 Mylex External DACs (EDACs). The DACs are plugged into the PCI slots on the motherboard, which are divided among three separate PCI buses.

The operating system, all executables and libraries, the master database, and swap space was contained in 1 HP Hot-swap 9Gbyte Ultra2 10krpm SCSI hard disk, attached to one of the embedded PCI SCSI controllers.

In the measured configuration, the database log drive consisted of 8 pairs of 9GB 10k RPM Ultra Wide hard disks attached to three Mylex EDACs through the embedded PCI SCSI controllers. The TPC-C database storage consisted of 192 hard drives, of which 48 drives were HP 4.2GB 10k RPM hot swap disks equally distributed across 2 PCI Disk Array Controllers (DACs) and 144 drives are HP 9.1GB 10k RPM hot swap disks equally distributed across 6 PCI Disk Array Controller (DACs). The drives on each DAC were equally distributed across the DAC SCSI channels. Each channel was striped and the channels were spanned using the AMI Series 438-H Super High Performance RAID Controller Utility. Controller write-back caching and read ahead were specifically disabled for the PCI DACs.

In the measured configuration NT's Disk Administrator showed 10 disk drives. They consist of one 9GB boot drive, one 54000MB disk drive containing the hardware mirrored sets of 9.1GB drives used for the log, six 208,248MB disk drives externalized by the DACs containing the 9.1GB disks, and two 104,136MB disks externalized by the DACs with 4.2GB disk drives. In the priced configuration a DAC of 9GB disks were substituted for a DAC of 4GB disks.

Eight logical disk drives were used to hold the TPC database. Protection against data loss from a failed drive was achieved by normal database level recovery and from the mirrored log drives.

Each of the clients is a HP NetServer LC 3 with dual Pentium II 400MHz, 512 Mbyte RAM, one 4 Gbyte SCSI hard disk, running Microsoft Windows NT Server 4.0.

A pair of HP Procurve 2400M switches were used to connect the server, clients and simulated users. The two switches were connected using two 100baseT links. The HP NetServer LXr 8000 and each of the clients used a single NetServer PCI network adapter card to provide a 100BaseT network interface to one of the switches. The clients were evenly distributed across the switches. The simulated users were connected to the switches' 10baseT links, and were evenly distributed across the two switches.

HP VGA displays were used on the server and each of the clients.

General Items

Test Sponsor

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

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

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 B.

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 A.

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.*

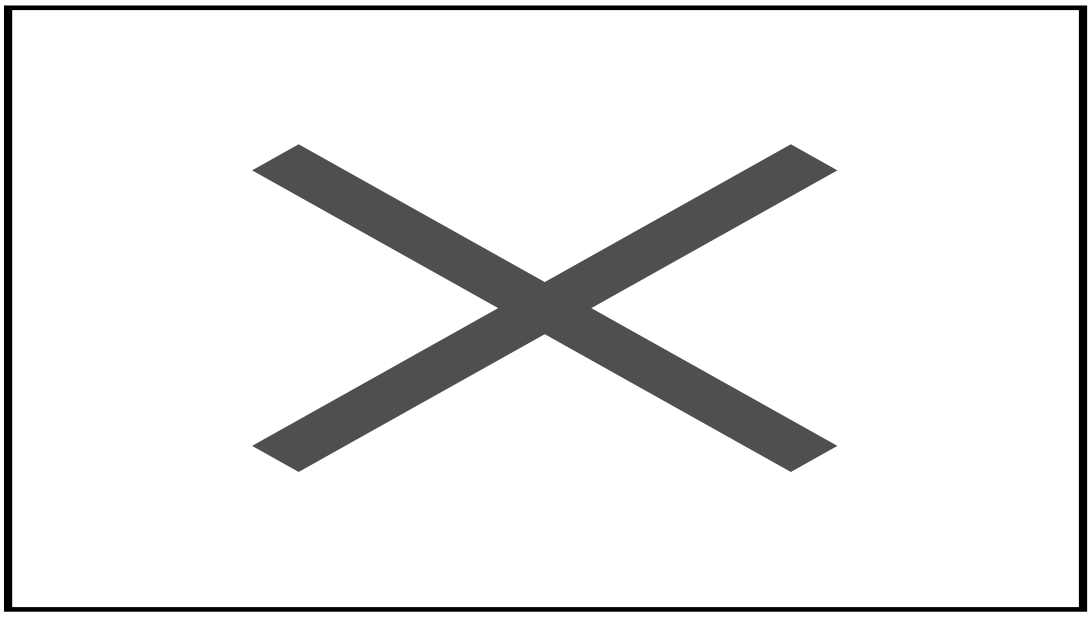
Appendix C contains all the database and operating system parameters used in this benchmark in addition to all the hardware configuration details.

Configuration Diagrams

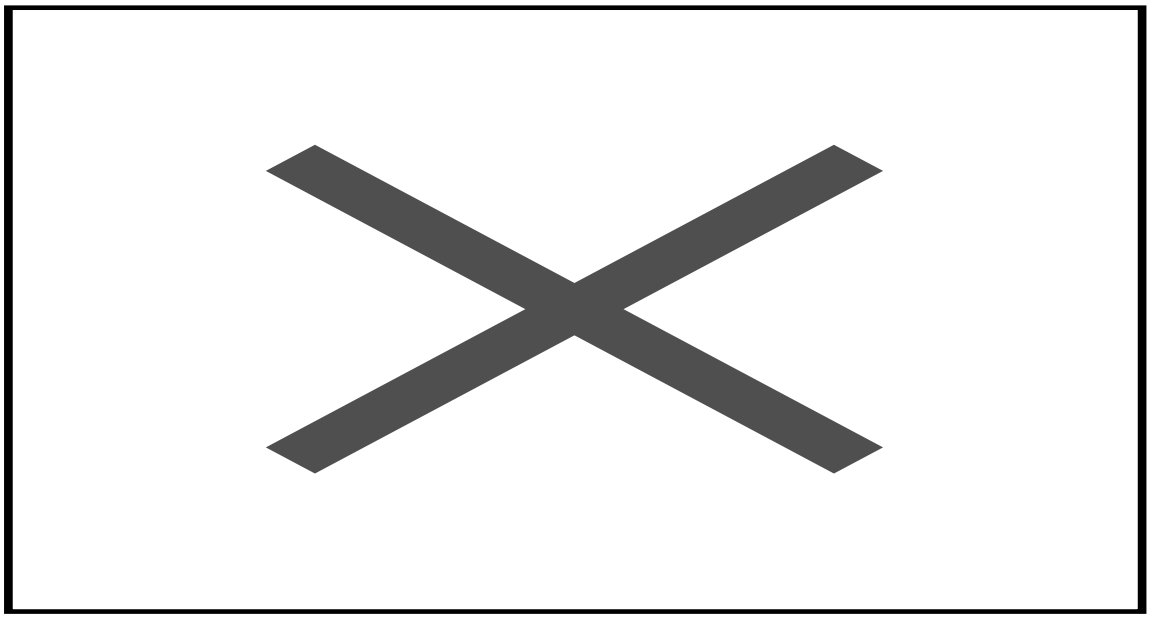
Diagrams of both the measured priced system must be provided, accompanied by a description of the differences.

The measured and priced client/server configurations are shown below

Measured Configuration



Priced Configuration



Chapter 1 Logical Database Design

1.1 Table Definitions

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

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

1.2 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 144 9Gbyte, and 48 4Gbyte Hot Swap disk drives for data, 16 9Gbyte drives for logs, and 1 9Gbyte drive for the operating system.

1.3 Insert and Delete Operations

It must be ascertained that insert 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.

1.4 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.

1.5 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.

Chapter 2 Transaction and Terminal Profiles

2.1 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.

2.2 Input/Output Screen Layout

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

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

2.3 Priced Terminal Feature Verification

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

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

2.4 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.

2.5 Transaction Statistics

The table below lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

Type	Item	Value
New Order	Home warehouse items	99.00%
	Remote warehouse items	1.00%
	Rolled back transactions	1.00%
	Average items per order	9.99
Payment	Home warehouse	84.94%
	Remote warehouse	15.06%
	Non primary key access	59.92%
Order Status	Non primary key access	59.79%
Delivery	Skipped transactions	0
Transaction Mix	New Order	44.87%
	Payment	43.02%
	Order Status	4.03%
	Delivery	4.03%
	Stock Level	4.05%

2.6 Queueing Mechanism

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

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

Chapter 3 Transaction and System Properties

3.1 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 a previously reported HP NetServer LXr 8000 system. (TPC-C disclosure dated Oct 12 1998). The durability tests of loss-of-system and loss-of-memory were performed on the measured configuration.

3.2 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.

3.2.1 COMMIT Transaction

Performed on a previously reported HP NetServer LXr 8000 system. (TPC-C disclosure dated Oct 12 1998)

3.2.2 ROLLBACK Transaction

Performed on a previously reported HP NetServer LXr 8000 system. (TPC-C disclosure dated Oct 12 1998)

3.3 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.

Performed on a previously reported HP NetServer LXr 8000 system. (TPC-C disclosure dated Oct 12 1998)

3.4 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.

Performed on a previously reported HP NetServer LXr 8000 system. (TPC-C disclosure dated Oct 12 1998)

3.5 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 whereas the Loss of Data and Loss of Log tests were performed on a previously reported HP NetServer LXr 8000 system. (TPC-C disclosure dated Oct 12 1998).

3.5.1 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 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 Memory test, the power switch on the HP NetServer LXr 8000 was depressed (turning off the system) while the benchmark was running.

Step 5: The recovery process was performed. 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 that 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.

Chapter 4 Scaling and Database Population

4.1 Database Layout

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

The measured (tested) and priced system have identical configurations. Both configurations have two Ultra-2 embedded SCSI PCI Disk controllers and 8 AMI Series 438-H Super High Performance RAID Controller 3-channel PCI Disk Array Controllers (DACs). These cards plugged into PCI slots on the motherboard.

1 HP Hot-swap 9Gbyte Ultra2 10krpm SCSI hard disk was attached to the first (A) of the two embedded PCI SCSI controllers. The drive was used for the operating system. One IDE CD-ROM was used.

For both the priced and measured configurations a total of 16 HP Hot-swap 9Gbyte 10krpm SCSI Hot Swap hard disks are used to supply growth space for the log. A Mylex EDAC was connected to the internal SCSI controller and had two sets of eight 9GB disk drives hardware mirrored.

In the measured configuration, 144 HP Hot-swap 9Gbyte 10krpm SCSI drives were attached to 6 of the AMI Series 438-H Super High Performance RAID Controller controllers and 48 HP Hot-swap 4Gbyte 10krpm SCSI drives were attached to the remaining controllers. Eight Hot Swap disks were placed in each HP Rack Storage 8. Each channel was striped using the AMI Series 438-H Super High Performance RAID Controller. Controller Utility and channel spanning was used. Controller write-back caching and read ahead were specifically disabled.

The priced configuration has 168 9.1GB drives and 24 4.2GB drives for data.

In the measured configuration, NT's Disk Administrator showed 10 disk drives. They consist of one 9GB boot drive, one 54000MB disk drive containing the hardware mirrored sets of 9.1GB drives used for the log, six 208,248 MB disk drives externalized by the AMI Series 438-H Super High Performance RAID Controller DACs containing the 9.1GB disks, and two 104,136MB disks externalized by the AMI Series 438-H Super High Performance RAID Controller DACs with 4.2GB disk drives.

4.2 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	Occurrences
Warehouse	1,885
District	18,850
Customer	56,550,000
History	56,550,000
Orders	56,550,000
New Orders	16,965,000
Order Line	497,998,198
Stock	100,000
Item	188,500,000

No rows were deleted for the benchmark runs.

4.3 180 Day Space

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

4.3.1 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:

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

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

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

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

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

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

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 56.7730 Gbyte (including mirror) to sustain the log for 8 hours. Space in the measured configuration available for the transaction log was 46.875 Gbyte with an additional 19.2GB available that can be configured, 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.

4.4 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 SQLServer 7.0 Enterprise Edition is a relational DBMS.

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

4.5 Database Mapping

The mapping of database partitions and replications must be described.

The database was divided into 2 file groups MSSQL70_cs_fg and MSSQL70_misc_fg each having eight files of the same size. Each of the eight logical data disk in the disk administrator has two partitions one of size 15900 MB to hold the MSSQL70_cs file and another of size 8000MB to hold the MSSQL70_misc file. The log was configured in a separate partition of size 54000MB.

Chapter 5 Performance Metrics and Response Time

5.1 Throughput

Measured tpmC® must be reported.

The measured tpmC® was 23,143.00

5.2 Response Times

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

Response Times	Average	90th %-ile	Maximum
New-Order	0.58s	0.92s	13.35s
Payment	0.39s	0.69s	3.16s
Order-Status	0.48s	0.80s	12.82s
Delivery (interactive portion)	0.22s	0.38s	1.80s
Delivery (deferred portion)	0.65s	0.99s	2.69s
Stock-Level	2.91s	3.61s	8.62s
Menu	0.28s	0.60s	2.63s

5.3 Keying and Think Times

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

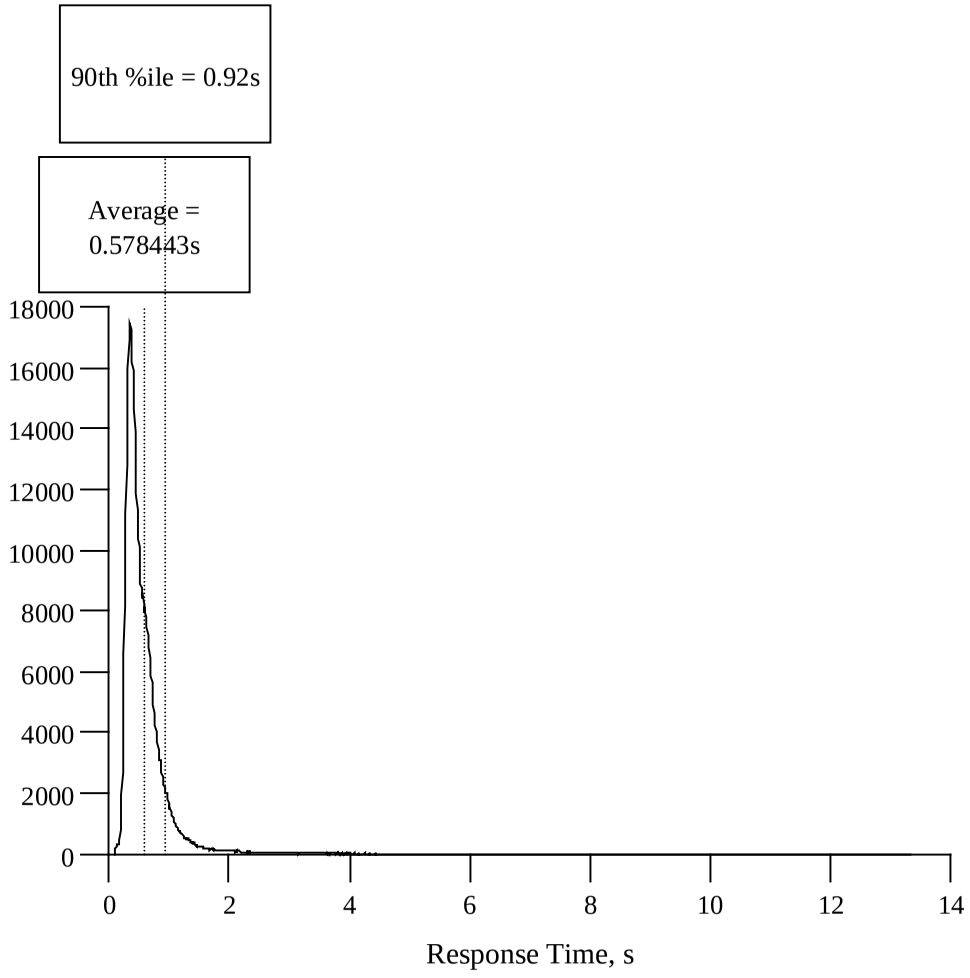
Keying Times	Minimum	Average	Maximum
New Order	18.02s	18.03s	18.18s
Payment	3.01s	3.02s	3.17s
Order Status	2.01s	2.02s	2.16s
Interactive Delivery	2.01s	2.02s	2.16s
Stock Level	2.01s	2.02s	2.17s

Think Times	Minimum	Average	Maximum
New Order	0.01s	12.15s	164.79s
Payment	0.01s	12.17s	176.39s
Order Status	0.01s	10.18s	117.89s
Interactive Delivery	0.01s	5.16s	55.92s
Stock Level	0.01s	5.2s	58.92s

5.4 Response Time Frequency

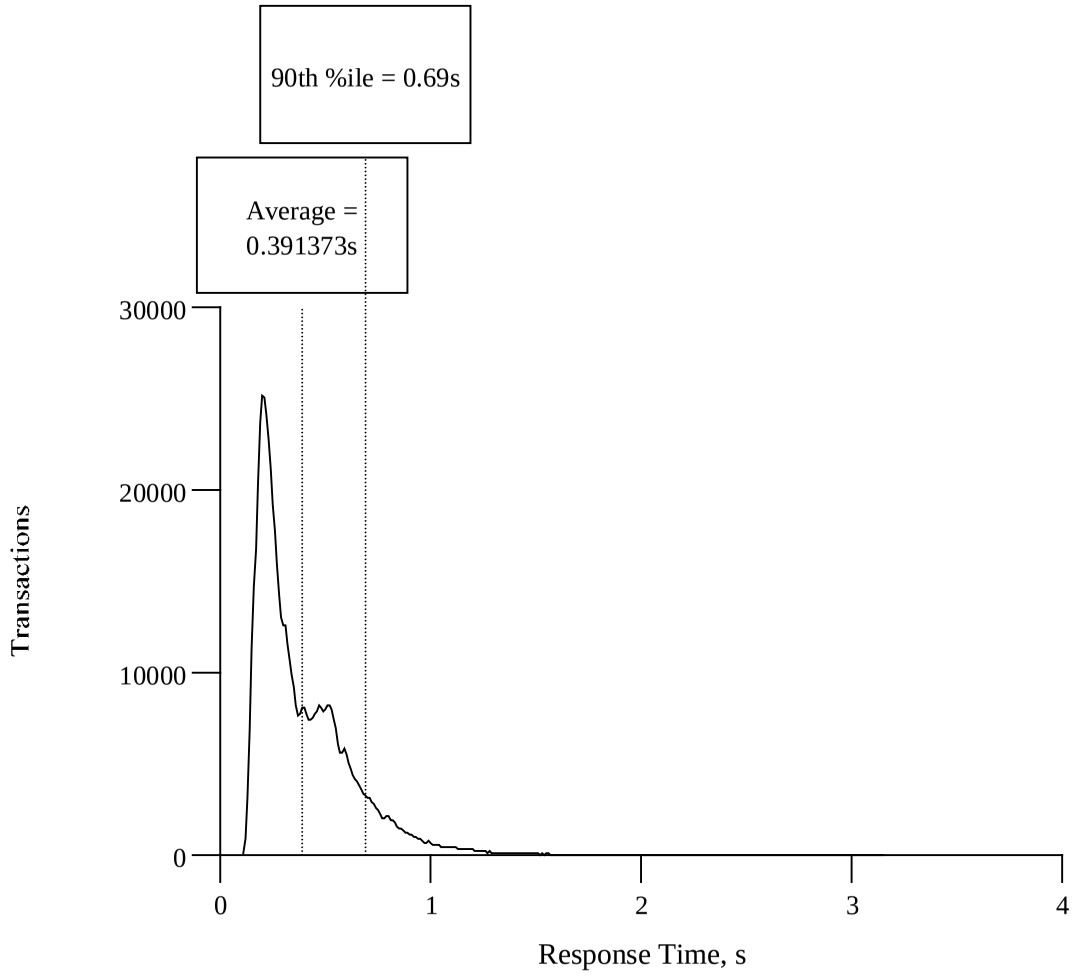
Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type. Keying Time frequency distribution curves (see Clause5.6.4) 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.

5.4.1 New Order Response Time



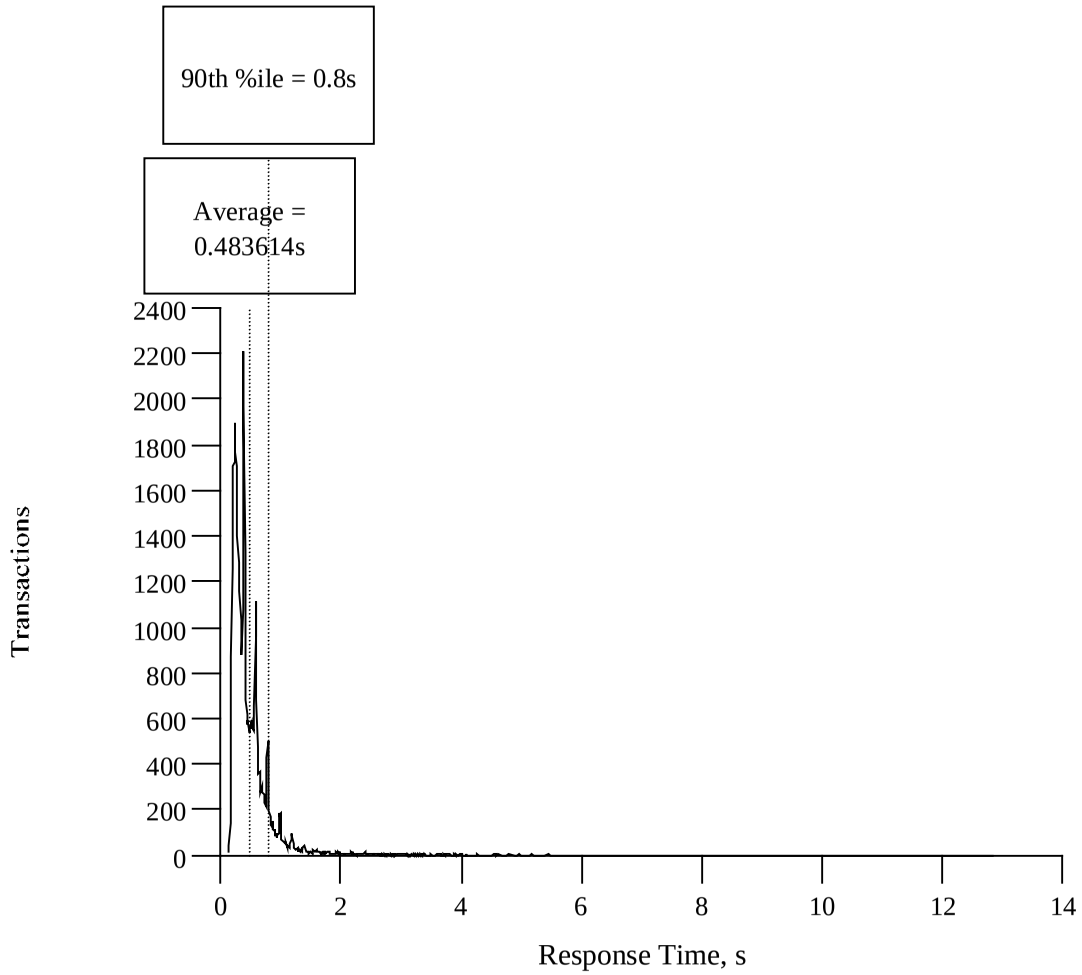
Response time frequency distribution for New Order transaction

5.4.2 Payment Response Time Distribution



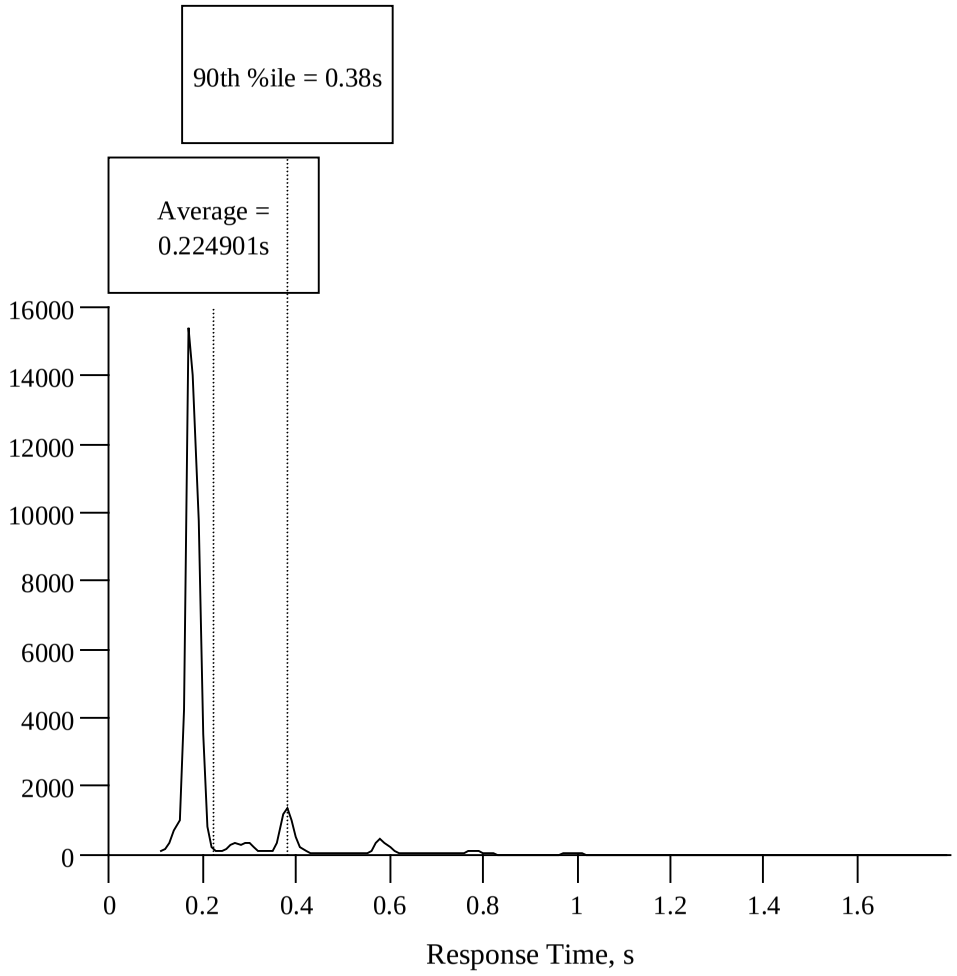
Response time frequency distribution for Payment transaction

5.4.3 Order Status Response Time



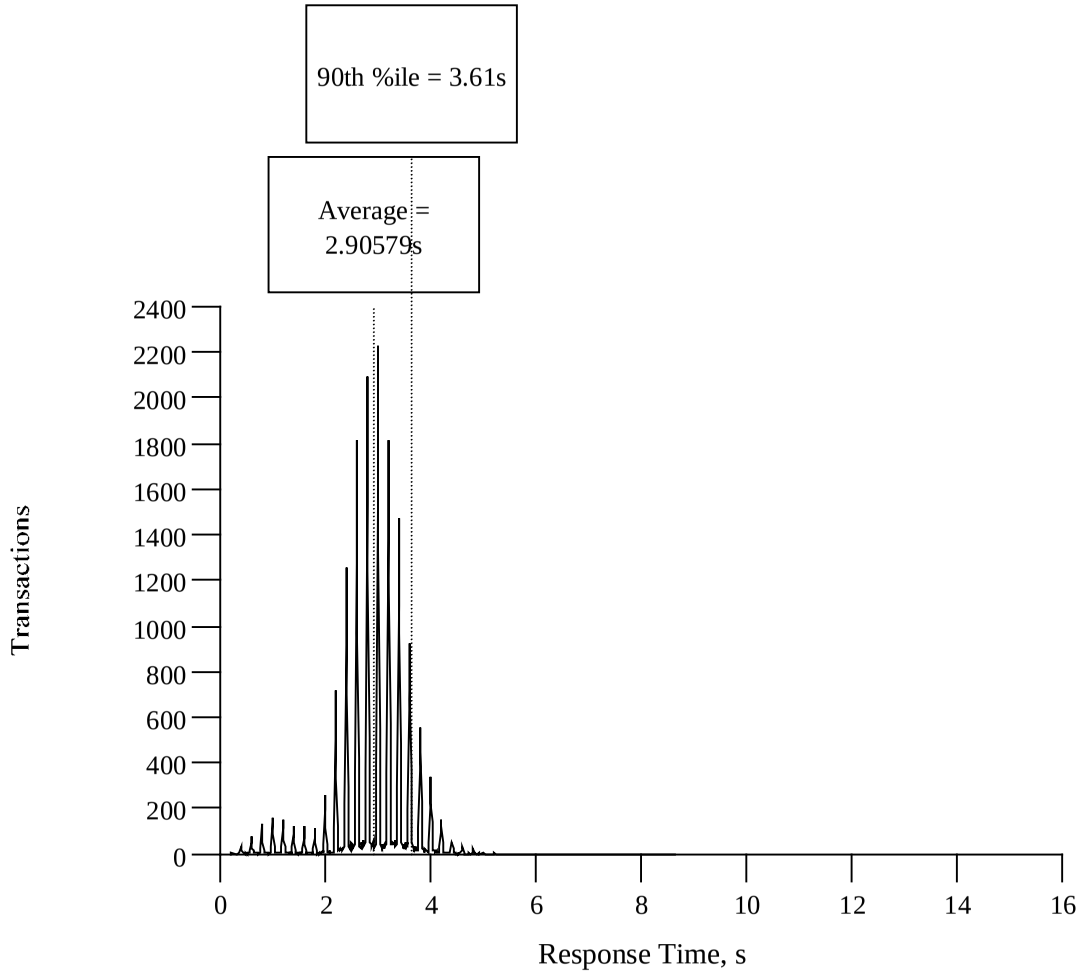
Response time frequency distribution for Order Status transaction

5.4.4 Delivery Response Time Distribution



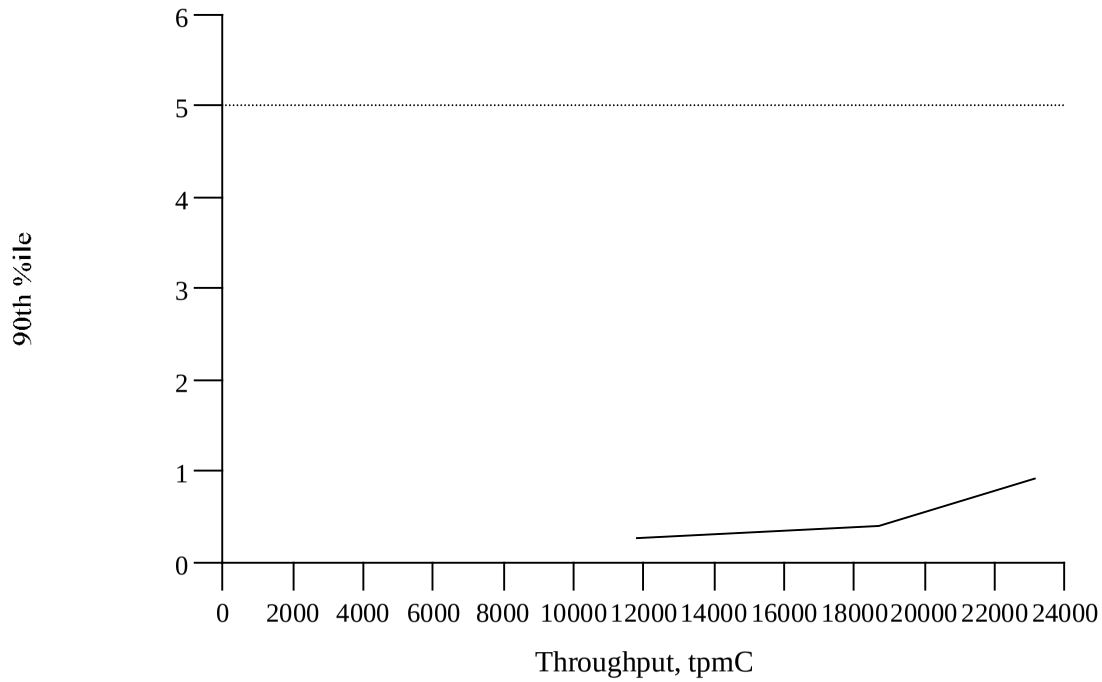
Response time frequency distribution for Delivery transaction

5.4.5 Stock Level Response Time



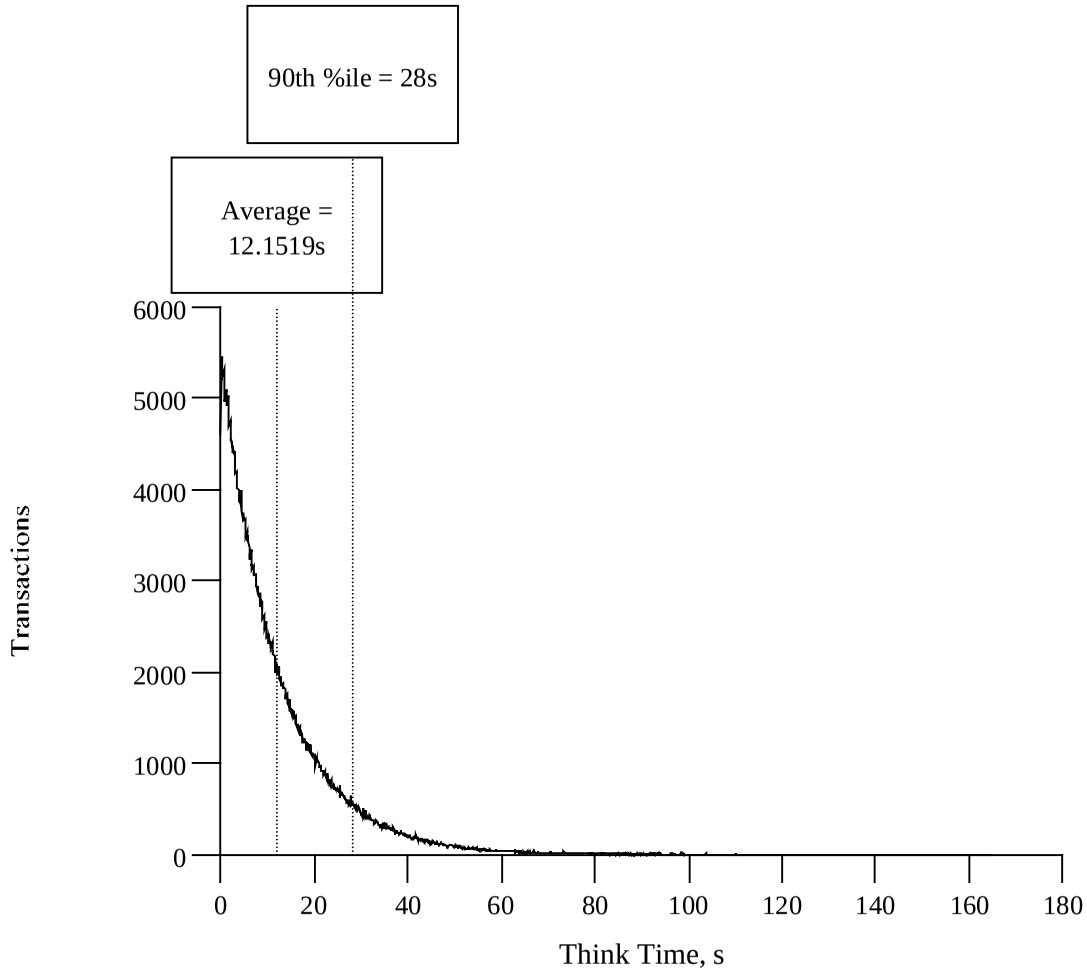
Response time frequency distribution for Stock Level transaction

5.4.6 Response Time Versus Throughput



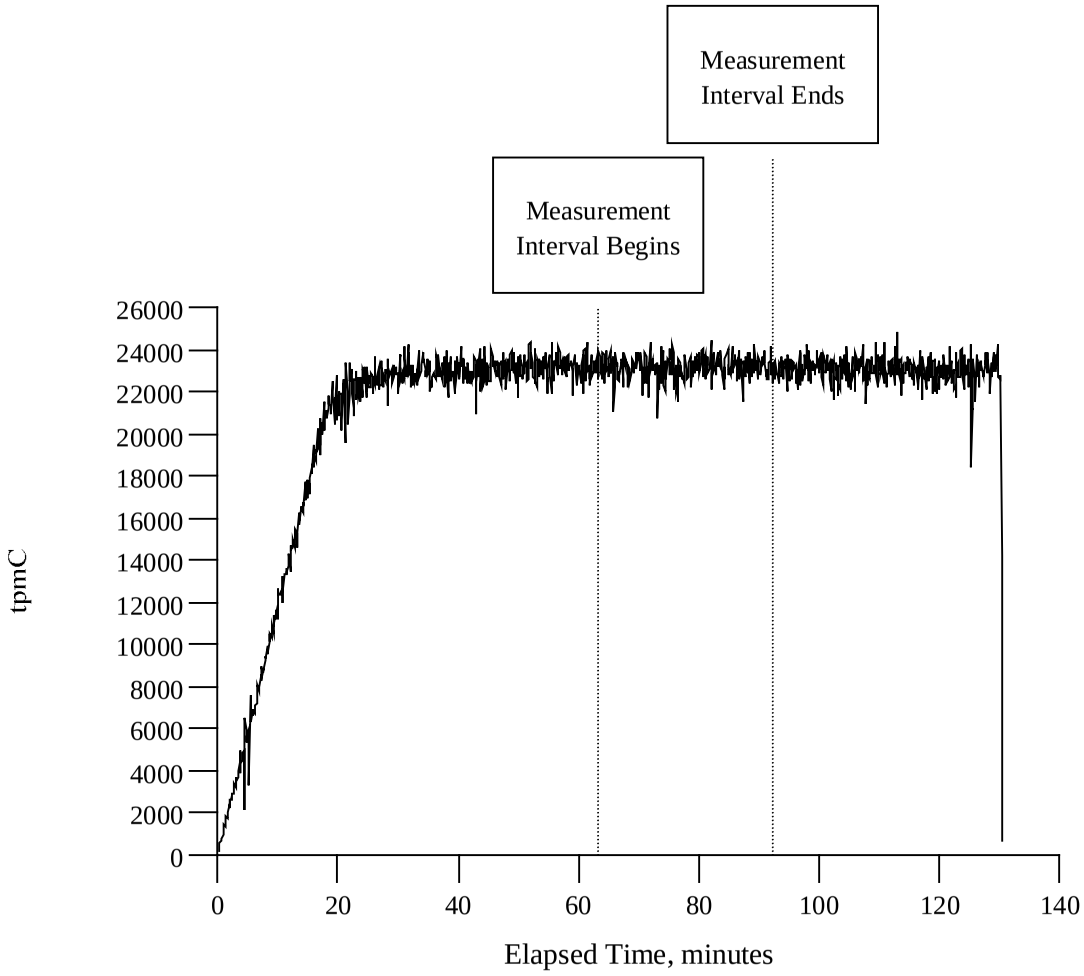
New Order response time versus Throughput

5.4.7 New Order Think Time Distribution



Think time frequency distribution for New Order transaction

5.4.8 Throughput Versus Time Distribution



Throughput of the New-Order transaction versus elapsed time

5.5 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.

5.6 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.

5.6.1 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.

5.6.2 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.

5.6.3 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 29 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.

5.7 Reproducibility

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

A second measurement achieved a throughput of 23,038.69 tpmC® during a 29-minute, steady state interval.

5.8 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 29 minutes.

5.9 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.

5.10 Transaction Mix

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

Type	Percentage
New Order	44.87%
Payment	43.02%
Order Status	4.03%
Delivery	4.03%
Stock Level	4.05%

5.11 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 3.1 contains the required items.

5.12 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 29 minutes. In accord with *5.5.2.2*, there is no checkpoint within the "guard zones" 1740/4=435 seconds from the beginning and end of the measurement interval.

Event	From (s)	To (s)	Duration (s)
Checkpoint	2940	3351	411
Measured Interval	3788	5528	1740
Checkpoint	4680	5046	366

Chapter 6 SUT, Driver and Communications Definition

6.1 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.

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

For this instance of the TPC-C benchmark, 9 driver and 4 client systems were used. The drivers emulated 18,850 users logged in to the clients. An overview of the benchmark software on the drivers, clients and server is shown in Section 6.2 below.

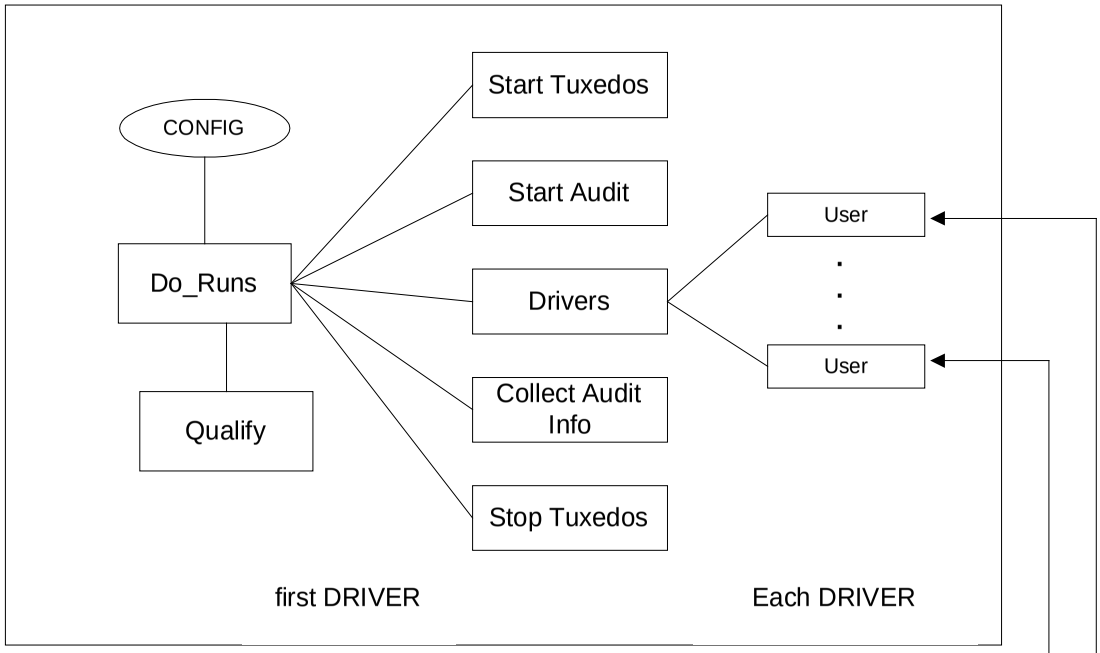
The benchmark is started with the do_runs command on the driver system. do_runs controls the overall execution of the benchmark. After reading a configuration file, do_runs starts the TUXEDO servers on the clients, collects pre-benchmark audit information and inserts a timestamp into a database audit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

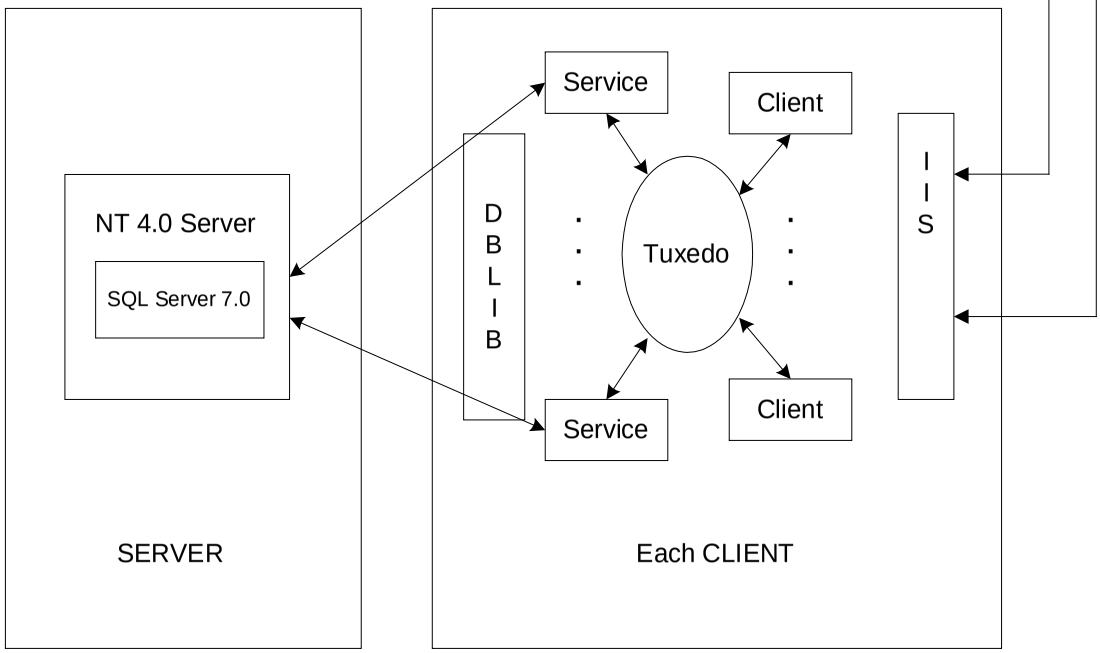
DRIVER is the heart of the benchmark software. It simulates users as they log in, execute transactions and view results. DRIVER collects response times for each transaction and saves them in a file for future analysis.

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.

6.2 Emulated Components





6.3 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.

Functional diagrams of the measured and priced systems are included in the "General Items" section at the beginning of this report. A description of the RTE and benchmark software is provided above.

6.4 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.

The "General Items" section includes diagrams of 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.

In both the measured and priced configurations, a pair of switches are used to provide connectivity. These switches are connected to each other by 2 100Mbps links. The server and the clients are all connected to these switches by a single 100Mbps link. The drivers (in the measured configuration) and the workstations and hubs (in the priced configuration) are connected to the switches using 10 Mbps links.

Chapter 7 Pricing

7.1 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.

7.2 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.

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:	23143 tpmC
Price per tpmC:	$26.87
Availability:	February 1, 1999

7.3 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.

7.4 Usage Pricing

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

The component pricing based on usage is shown below:

Microsoft SQLServer 7.0 Enterprise Edition was priced for unlimited number of users.

Chapter 8 Audit

8.1 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.
2229 Benita Drive, Suite 101
Rancho Cordova, CA 95670
U.S.A.
Phone: 916 635-2822
Fax: 916 858-0109

The auditor was Tom Sawyer. 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
Enterprise NetServer Division
10955 Tantau Avenue
Cupertino, CA 95014-0770 USA
Attn:Neil MacDonald, MS 45NUH

PERFORMANCE METRICS INC.
TPC Certified Auditors

December 7, 1998

Neil MacDonald
Network Server Division
Hewlett-Packard Company
10955 Tantau Avenue
Cupertino, CA 95014

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

Platform:	Hewlett-Packard NetServer LXr 8000
Database Manager:	Microsoft SQL Server Enterprise Edition 7.0
Operating System:	Microsoft Windows NT Server Enterprise Edition 4.0 (SP3)
Transaction Manager:	BEA TUXEDO CFS 6.4 for NT

Server: Hewlett-Packard NetServer LXr 8000				
CPU's	Memory	Disks	90% Response	tpmC
4 Pentium II Xeon @ 450 Mhz	Main: 4 GB Cache: 2MB	161 @ 9 GB 48 @ 4 GB	0.92 sec	23,143.00
4 Clients: Hewlett-Packard NetServer LC 3 (each)				
2 Pentium II @ 400 MHz	Main: 512 MB Cache: 512 KB	1 @ 4 GB	Na	na

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark. The following attributes of the benchmark were given special attention:

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 1,885 warehouses.
- The ACID properties were met.

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

- The ACID tests were performed on a previously reported system that used the same hardware and binaries. The Durability tests for loss-of-system and loss-of-memory were performed on the measured configuration
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was present on the tested system.
- Eight hours of growth space for the dynamic tables was present on the tested system.
- The data for the 180 day space calculation was verified; 24 4GB disks were upgraded to 9GB disks in the priced configuration. I verified that the substitution would not affect performance.
- The steady state portion of the test was 29 minutes.
- One checkpoint was taken before the measured interval.
- One checkpoint was taken during the measured interval.
- The checkpoints were verified to be clear of the guard zone.
- The system pricing was checked for major components and maintenance.

Auditor Notes:

None.

Sincerely,
Tom Sawyer
Auditor

Appendix A Application Source

A.1 Client Front End

File: 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
```

File: delirpt.c

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

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

```
#define LOGFILE_READ_EOF 0
                //check log file flag return current state
#define LOGFILE_CLEAR_EOF1
                //clear end of log file flag
#define LOGFILE_SET_EOF          2
                        //set flag end of log file reached

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

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

typedef struct _RPTLINE
{
        SYSTEMTIME      start;
                                //delilog report line start time
        SYSTEMTIME      end;
                                //delilog report line end time
        int                     response;
                                        //delilog report line time delivery
took in milliseconds
        int                     w_id;
                                        //delilog report line warehouse id
for delivery
        int                     o_carrier_id;
                                //delilog report line carier id for delivery
        int                     items[10];
                                        //delilog report line delivery line
items
} RPTLINE, *PRPTLINE;

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

```
int                             versionMS = 4;
                //delirpt version
int                             versionMM = 0;
int                             versionLS = 0;
int                             iReport;
                //delirpt report to process
int                             iStartTime;
                        //begin times to accept for report
int                             iEndTime;
                        //end times to accept for report
FILE                    *fpLog;
                //log file stream

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


/* FUNCTION: int main(int argc, char *argv[])
 *
 * PURPOSE:     This function is the beginning execution point for the
delivery executable.
 *
```

```
 * ARGUMENTS:    int             argc    number of command line arguments
passed to delivery
 *                               char    *argv[]  array of command line
argument pointers
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

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

        PrintHeader();

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

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

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

        Restore();

        return;
}

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

static int Init(void)
{
        int iError;

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

/* FUNCTION: static void Restore(void)
 *
 * PURPOSE:      This function cleans up the delirpt application before
termination.
 *
 * ARGUMENTS:    None
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

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

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

```
static int DoReport(void)
{
        int iRc;

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

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

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

        ResetLogFile();

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

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

        return ERR_SUCCESS;
}

/* FUNCTION: int Percentile90th(void)
 *
 * PURPOSE:             This function processes the 90th percentile report.
```

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

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

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

        ResetLogFile();

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

        iTotalBuckets = iMaxSeconds + 1;

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

        iBucketSize = iTotalBuckets * sizeof(short);

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

        ZeroMemory(psBuckets, iBucketSize);

        iTotal = 0;

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

        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;
        char    szDelivery[128];
        int             i;
        int             items[10];

        ResetLogFile();

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

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

        printf("\n");

        return ERR_SUCCESS;
}

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

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

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

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

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

        return TRUE;
```

```
}

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

static int OpenLogFile(void)
{
        fpLog = fopen("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 iOperation   requested operation this can be:
 *
        LOGFILE_READ_EOF check log file flag return current state
 *
        LOGFILE_CLEAR_EOF        clear end of log file flag
 *
        LOGFILE_SET_EOF          set flag end of log file reached
 *
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

static BOOL LogEOF(int iOperation)
```

```
{
        static BOOL bEOF;

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

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

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

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

        }

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

        return FALSE;
}

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

static BOOL ParseDate(char *szDate, LPSYSTEMTIME pTime)
{
        if ( !isdigit(*szDate) || !isdigit(*(szDate+1)) || *(szDate+2) != '/'
||
```

```
                !isdigit(*(szDate+3)) || !isdigit(*(szDate+4)) || *(szDate+5)
!= '/' ||
                !isdigit(*(szDate+6)) || !isdigit(*(szDate+7)) )
                return TRUE;

        pTime->wYear = atoi(szDate);

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

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

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

        return FALSE;
}

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

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

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

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

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

        return FALSE;
}

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

static void ErrorMessage(int iError)
{
        int i;

        static SERRORMSG errorMsgs[] =
        {
                {       ERR_SUCCESS,
                "Success, no error."
                                                        },
```

```
        {       ERR_CANNOT_OPEN_RESULTS_FILE,
"Cannot open delivery results 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:   int             argc    number of command line arguments
passed to delivery
 *                              char    *argv[]  array of command line
argument pointers
 *
 * RETURNS:             BOOL    FALSE   parameter read successfull
 *                                      TRUE    user has requested
parameter information screen be displayed.
 *
 * COMMENTS:    None
 *
 */

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

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

        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

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

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

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

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

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

        return;
}

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

static BOOL IsNumeric(char *ptr)
{
        if ( *ptr == 0 )
                return FALSE;

        while( *ptr && isdigit(*ptr) )
                ptr++;
        if ( !*ptr || *ptr == ',' )
                return TRUE;
        else
                return FALSE;
}
```

File: delisrv.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>
        HENV    henv;
#else
        #define DBNTWIN32
        #include <sqlfront.h>
        #include <sqldb.h>
#endif

#include "delisrv.h"

char                                    szServer[32];
        //SQL server name
```

```
char                                    szDatabase[32];
        //tpcc database name
char                                    szUser[32];
                //user name
char                                    szPassword[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_SECTION        WriteLogCriticalSection; //critical section for
delivery write log
CRITICAL_SECTION        DeliveryCriticalSection; //critical section for
delivery transactions cache
static  LPTSTR                  lpszPipeName = TEXT("\\\\.\\pipe\\DELISRV");
        //delivery pipe name

HANDLE                                  hPipe           =
INVALID_HANDLE_VALUE;   //delivery pipe handle
HANDLE                                  hComPort = INVALID_HANDLE_VALUE;
        //delivery pipe completion port handle.

BOOL                                    bDone;
                        //delivery executable termination request flag
BOOL                                    bFlush;
                        //Flush delivery log info when written.

LPDELIVERY_PACKET               pDeliveryCache;

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:   int             argc    number of command line arguments
passed to delivery
 *                              char    *argv[]  array of command line
argument pointers
 *
 * RETURNS:             None
 *
 * COMMENTS:    None
 *
 */

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

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

        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

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

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

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

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

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

        return;
}

/* FUNCTION: int RunDelivery(void)
 *
 * PURPOSE:      This function executes the main delivery executable loop.
 *
 * ARGUMENTS:    None
 *
 * RETURNS:              int     ERR_CANNOT_OPEN_PIPE            cannot open
named pipe
 *                                      ERR_CANNOT_CREATE_THREAD cannot
create required threads
 *                                      ERR_SUCCESS
                successfull no error
 *
 *
 * COMMENTS:     None
 *
 */

static int RunDelivery(void)
{
        SECURITY_ATTRIBUTES     sa;
        int                                     i;

        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:              int     iError          Error code if unsuccessfull
 *                                       ERR_SUCCESS     No error successfull
code
 *
```

```
 *
 * COMMENTS:     None
 *
 */

static int Init(void)
{
        int     iError;

        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)
{
        int     iret, 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:    int             iError  error id to be displayed
 *
 * RETURNS:              None
 *
```

```
 * COMMENTS:     None
 *
 */

static void ErrorMessage(int iError)
{
        int i;

        static SERRORMSG errorMsgs[] =
        {
                {       ERR_SUCCESS,
                "Success, no error."
                },
                {       ERR_CANNOT_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:    int             argc    number of command line arguments
passed to delivery
 *                               char    *argv[]  array of command line
argument pointers
 *
 * RETURNS:              BOOL    FALSE   parameter read successfull
```

```
 *                                                TRUE    user has requested
parameter information screen be displayed.
 *
 * COMMENTS:     None
 *
 */

static BOOL GetParameters(int argc, char *argv[])
{
        int i;


        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:              int     ERR_REGISTRY_NOT_SETUP           registry not
setup tpcc.exe needs to be run
 *
                         to setup registry.
 *                                    ERR_SUCCESS
                Registry read Successfull, no error
 *
 *
 * COMMENTS:     None
 */

static int ReadRegistrySettings(void)
{
        HKEY    hKey;
        DWORD   size;
        DWORD   type;
        char    szTmp[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    *ptr    dummy 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    *ptr    dummy argument passed in though thread
manager, unused NULL.
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

static void DeliveryHandler( void *ptr )
{
        int     i;
        int     size;
        int     iError;

        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    *ptr    dummy 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 )
{
        int                             size;
        int                             key;
        LPOVERLAPPED    pov;
        DELIVERY        delivery;
        int                             iError;

        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               *dbproc                 DBPROCESS id
pointer
 *                              int                             severity
        severity of error
 *                              int                             dberr
                error id
 *                              int                             oserr
                operating system specific error code
 *                              char                    *dberrstr
        printable error description of dberr
 *                              char                    *oserrstr
        printable error description of oserr
 *
 * RETURNS:              int                             INT_CONTINUE
        continue 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               *dbproc                 DBPROCESS id
pointer
 *                              DBINT                   msgno
        message number
 *                              int                             msgstate
        message state
 *                              int                             severity
        message severity
 *                              char                    *msgtext
        printable message description
 *
 * RETURNS:              int                             INT_CONTINUE
        continue if error is SQLETIME else INT_CANCEL action
 *                                                              INT_CANCEL
                cancel 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               **dbproc pointer to returned DBPROCESS
 *                               char                    *server         SQL
server name
 *                               char                    *database       SQL
server database
 *                               char                    *user           user
name
 *                               char                    *password       user
password
 *                               int                             *spid
        pointer to returned spid
 *
 * RETURNS:              BOOL    FALSE   if successfull
 *                                       TRUE    if an error occurs
 *
 * COMMENTS:     None
 *
 */

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

                RETCODE         rc;
                char            buffer[30];

                *dbproc = (DBPROCESS *)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:   LPDELIVERY      pDelivery       Pointer 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                             *pElapsed       pointer to
int variable to receive calculated elapsed
 *
                time in milliseconds.
 *                              LPSYSTEMTIME    lpBegin         Pointer to
system time structure containing
 *
                transaction beginning time.
 *                              LPSYSTEMTIME    lpEnd           Pointer to
system time structure containing
 *
                transaction ending time.
 * RETURNS:             None
 *
 * COMMENTS:     None
 *
 */

static void CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME lpBegin,
LPSYSTEMTIME lpEnd)
{
        int                     beginSeconds;
        int                     endSeconds;


        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        *pDelivery              Pointer to delivery
transaction structure
 *
 * RETURNS:              int     ERR_DBGETDATA_FAILED            Delivery get
data operation failed.
 *                                      ERR_SUCCESS
                Delivery successfull, no error
 *
 *
 * COMMENTS:     None
 *
 */

#ifdef USE_ODBC
        static int SQLDelivery(DELIVERY *pDelivery)
        {
                int             i;
                int             deadlock_count;
                SDWORD  iLength[10];
                BOOL    bDeadlock;

                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)
        {
                RETCODE rc;
                int             i;
                int             deadlock_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               *dbproc         DBPROCESS to check
 *
 * RETURNS:             BOOL    FALSE                           No lock
condition present
 *                                                      TRUE
        Lock 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:              int     ERR_REGISTRY_NOT_SETUP
        Registry not setup.
 *                                      ERR_CANNOT_CREATE_RESULTS_FILE
        Cannot create results log file.
 *                                      ERR_SUCCESS
                        Log file successfully opened
 *
 *
 * COMMENTS:     None
 *
 */

static int OpenLogFile(void)
{
        HKEY    hKey;
        BOOL    bRc;
        BYTE    szTmp[256];
        char    szKey[256];
        char    szLogPath[256];
        DWORD   size;
        DWORD   sv;
        int             len;
        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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *                               void            *uPtr   returned
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:    DBRPOCESS       dbproc  ODBC 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:    PDBPROCESS      dbproc          pointer to odbc
dbprocess structure
         *                               UWORD   ipar            Parameter
number, ordered sequentially left to right, starting at 1.
         *                               SWORD   fParamType      The type of
the parameter.
         *                               SWORD   fCType          The C data
type of the parameter.
         *                               SWORD   fSqlType The SQL data type of
the parameter.
         *                               UDWORD  cbColDef The precision of the
column or expression
         *
        of the corresponding parameter marker.
         *                               SWORD   ibScale         The scale of
the column or expression of the corresponding
         *
        parameter marker.
         *                               PTR             rgbValue A pointer to
a buffer for the parameter s data.
         *                               SDWORD  cbValueMax      Maximum
length of the rgbValue buffer.
         *                               void            *uPtr   returned
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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *
         * RETURNS:              none
         *
         * COMMENTS:     none
         *
         */

        void ODBCError(PDBPROCESS dbproc)
        {
                SDWORD          lNativeError;
                char            szState[6];
                char            szMsg[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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *                               char            *szStatement    sql
stored procedure statement to be executed.
         *
         * RETURNS:              none
         *
         * COMMENTS:     none
         *
         */

        BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement)
        {
                RETCODE         rc;

                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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *                               UWORD   icol
        Column number of result data, ordered sequentially left to right,
starting at 1.
```

```
         *                              SWORD    fCType                    The
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
         *                              PTR              rgbValue
        Pointer to storage for the data. If rgbValue is a null pointer, the
         *
                driver unbinds the column.
         *                              SDWORD   cbValueMax
        Maximum length of the rgbValue buffer. For character data, rgbValue
         *
                must also include space for the null-termination byte.
         *                              SDWORD *piLength         Pointer 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)
        {
                RETCODE          rc;

                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:    DBRPOCESS        dbproc   ODBC 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:    DBRPOCESS        dbproc   ODBC 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
```

File: 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 AVAILABLE               0
        //queue array element available
#define WRITE_LOCKED    1
        //queue array element is being written to
#define READ_LOCKED             2
        //queue array element is begin read
#define  INUSE                  4
        //queue array element has information stored in it

#define CTRL_C                  3
        //<Ctrl> C, exit key code

#define DEFCLPACKSIZE                                   4096    //default DB
Library SQL Connection pack size

#define  ERR_SUCCESS                                            0
        //Success, no error.
#define ERR_CANNOT_CREATE_THREAD        1000    //Cannot create thread.
#define ERR_DBGETDATA_FAILED                    1001    //Get data failed.
#define ERR_REGISTRY_NOT_SETUP                  1002    //Registry not setup
for tpcc.
#define ERR_CANNOT_ACCESS_DELIVERY_FN   1003    //Cannot access ReadDelivery
cache.
#define ERR_CANNOT_ACCESS_REGISTRY              1004    //Cannot access
registry key TPCC.
#define ERR_CANNOT_CREATE_RESULTS_FILE  1005    //Cannot create results file.
#define ERR_CANNOT_OPEN_PIPE                    1006    //Cannot open
delivery pipe.
#define ERR_READ_PIPE                                   1007    //Error
reading pipe
#define ERR_INSUFFICIENT_MEMORY                 1008    //insufficient
memory
#define ERR_ODBC_SQLALLOCENV                    1009    //Cannot allocated
ODBC env handle
#define ERR_SQL_ATTR_ODBC_VERSION       1010    //Cannot set ODBC version
#define ERR_SQL_ATTR_CONNECTION_POOLING 1011    //Cannot set Connection
Pooling

typedef struct _DELIVERY_TRANSACTION
{
        SYSTEMTIME      queue;                  //time delivery transaction
queued
        short           w_id;                   //delivery warehouse
        short           o_carrier_id;   //carrier id
} DELIVERY_TRANSACTION;

typedef DELIVERY_TRANSACTION *LPDELIVERY_TRANSACTION;     //pointer to
delivery transaction queue

typedef struct _DELIVERY_PACKET
{
        BOOL                                    bInUse;
        //entry current in use
        OVERLAPPED                              ov;
        //pipe io overlapped structure
        DELIVERY_TRANSACTION    trans;                  //delivery
transaction information
} DELIVERY_PACKET, *LPDELIVERY_PACKET;

typedef struct _SERRORMSG
{
        int             iError;                 //error mesage id
        char    szMsg[80];              //error message
} SERRORMSG;
```

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

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

//delivery transaction structure
typedef struct DELIVERY
{
        short           w_id;                   //warehouse id
        short           o_carrier_id;   //carrier id
        int                     spid;                   //db library spid
        long            o_id[10];               //returned delivery
transaction ids
        DBPROCESS       *dbproc;        //db library DBPROCESS pointer
        SYSTEMTIME      queue;                  //delivery transaction queue
time
        SYSTEMTIME      trans_end;              //delivery transaction
finished time
} DELIVERY;

typedef DELIVERY *LPDELIVERY;    //pointer to delivery structure

//function prototypes
void                    main(int argc, char *argv[]);
static  void    cls(void);
static  int             RunDelivery(void);
static  void    QuitStatus(void);
static  void    AnimateWait1(void);
static  void    AnimateWait(void);
static  int             Init(void);
static  void    Restore(void);
static  void    ErrorMessage(int iError);
static  BOOL    GetParameters(int argc, char *argv[]);
static  void    PrintParameters(void);
static  void    PrintHeader(void);
static  int             ReadRegistrySettings(void);
static  void    CheckKey(void *ptr);
static  void    DeliveryHandler( void *ptr );
static  void    DeliveryThread( void *ptr );

#ifndef USE_ODBC
        static  int             err_handler(DBPROCESS *dbproc, int severity,
int dberr, int oserr, char *dberrstr, char *oserrstr);
#endif

#ifdef USE_ODBC
        #define DBINT    int
#endif

static  int             msg_handler(DBPROCESS *dbproc, DBINT msgno, int
msgstate, int severity, char *msgtext);
static  BOOL    SQLOpenConnection(DBPROCESS **dbproc, char *server, char
*database, char *user, char *password, int *spid);
static  void    WriteLog(LPDELIVERY pDelivery);
static  void    CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME lpBegin,
LPSYSTEMTIME lpEnd);
static  int             SQLDelivery(DELIVERY *pDelivery);
static  BOOL    SQLDetectDeadlock(DBPROCESS *dbproc);
static  BOOL    ReadDeliveryInfo(short *w_id, short *o_carrier_id);
static  BOOL    PostDeliveryInfo(short w_id, short o_carrier_id);
static  int             OpenLogFile(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
```

File: dll.mak

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

OUT_PATH= .

ALL:    $(OUT_PATH)\. dlls

$(OUT_PATH)\.:
        if not exist $(OUT_PATH) md $(OUT_PATH)

#dlls:tuxedo_process_dll tuxedo_threads_dll
dlls:tuxedo_threads_dll

original_dll:
        cd original
        nmake CFG=$(CFG) /$(MAKEFLAGS) /f original.mak
        cd ..

tuxedo_process_dll:
        cd tuxedo_process
        nmake CFG=$(CFG) /$(MAKEFLAGS) /f tuxedo_process.mak
        cd ..

tuxedo_threads_dll:
        cd tuxedo_threads
        nmake CFG=$(CFG) /$(MAKEFLAGS) /f tuxedo_threads.mak
        cd ..
```

File: 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  *pECB   passed in structure pointer
from inetsrv.
 *                                int                                iError
                id of error message
 *                                int                                iErrorType
                error type, ERR_TYPE_SQL, ERR_TYPE_DBLIB, or ERR_TYPE_WEBDLL
 *                                int                                iTermId
                terminal id from browser
 *                                int                                iSyncid
                sync id from browser
 *                                char *                    szMsg
        optional 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;
}
```

File: error.h

```
#ifndef ERROR_H_INCLUDED
#define ERROR_H_INCLUDED

extern TERM Term;

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

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_ID                                                 1
                //expected abort record in txnRecord
#define ERR_TYPE_DELIVERY_POST                                  2
        //expected delivery post failed
#define ERR_TYPE_WEBDLL                                                 3
                //tpcc web generated error
#define ERR_TYPE_SQL                                                    4
                //sql server generated error
#define ERR_TYPE_DBLIB                                                  5
                //dblib generated error
#define ERR_TYPE_ODBC                                                   6
                //odbc generated error
#define ERR_TYPE_SOCKET                                                 7
                //error on communication socket client rte only
#define ERR_TYPE_DEADLOCK                                       8
        //dblib and odbc only deadlock condition

#define ERR_SUCCESS
        1000    //"Success, no error.
#define ERR_COMMAND_UNDEFINED                                   1001
        //"Command undefined.
#define ERR_NOT_IMPLEMENTED_YET                                 1002
        //"Not Implemented Yet.
#define ERR_CANNOT_INIT_TERMINAL                        1003    //"Cannot
initialize client connection.
#define ERR_OUT_OF_MEMORY                                               1004
        //"insufficient memory.
#define ERR_NEW_ORDER_NOT_PROCESSED                             1005
        //"Cannot process new Order form.
#define ERR_PAYMENT_NOT_PROCESSED                       1006    //"Cannot
process payment form.
#define ERR_NO_SERVER_SPECIFIED                                 1007
        //"No Server name specified.
#define ERR_ORDER_STATUS_NOT_PROCESSED                  1008    //"Cannot
process order status form.
#define ERR_W_ID_INVALID                                        1009
        //"Invalid Warehouse ID.
#define ERR_CAN_NOT_SET_MAX_CONNECTIONS                 1010
        //"Insufficient memory to allocate # connections.
#define ERR_NOSUCH_CUSTOMER                                             1011
        //"No such customer.
```

```
#define ERR_D_ID_INVALID                                    1012
        //"Invalid District ID Must be 1 to 10.
#define ERR_MAX_CONNECT_PARAM                               1013
        //"Max client connections exceeded, run install to increase.
#define ERR_INVALID_SYNC_CONNECTION                         1014
        //"Invalid Terminal Sync ID.
#define ERR_INVALID_TERMID                                          1015
        //"Invalid Terminal ID.
#define ERR_PAYMENT_INVALID_CUSTOMER                1016    //"Payment
Form, No such Customer.
#define ERR_SQL_OPEN_CONNECTION                             1017
        //"SQLOpenConnection API Failed.
#define ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY    1018    //"Stock Level
missing Threshold key "TT*".
#define ERR_STOCKLEVEL_THRESHOLD_INVALID        1019    //"Stock Level
Threshold invalid data type range = 1 - 99.
#define ERR_STOCKLEVEL_THRESHOLD_RANGE              1020    //"Stock
Level Threshold out of range, range must be 1 - 99.
#define ERR_STOCKLEVEL_NOT_PROCESSED                1021    //"Stock
Level not processed.
#define ERR_NEWORDER_FORM_MISSING_DID               1022    //"New Order
missing District key "DID*".
#define ERR_NEWORDER_DISTRICT_INVALID               1023    //"New Order
District ID Invalid range 1 - 10.
#define ERR_NEWORDER_DISTRICT_RANGE                 1024
        //"New Order District ID out of Range. Range = 1 - 10.
#define ERR_NEWORDER_CUSTOMER_KEY                   1025    //"New Order
missing Customer key "CID*".
#define ERR_NEWORDER_CUSTOMER_INVALID               1026    //"New Order
customer id invalid data type, range = 1 to 3000.
#define ERR_NEWORDER_CUSTOMER_RANGE                 1027
        //"New Order customer id out of range, range = 1 to 3000.
#define ERR_NEWORDER_MISSING_IID_KEY                1028    //"New Order
missing Item Id key "IID*".
#define ERR_NEWORDER_ITEM_BLANK_LINES               1029    //"New Order
blank order lines all orders must be continuous.
#define ERR_NEWORDER_ITEMID_INVALID                 1030
        //"New Order Item Id is wrong data type, must be numeric.
#define ERR_NEWORDER_MISSING_SUPPW_KEY              1031    //"New Order
missing Supp_W key "SP##*".
#define ERR_NEWORDER_SUPPW_INVALID                  1032
        //"New Order Supp_W invalid data type must be numeric.
#define ERR_NEWORDER_MISSING_QTY_KEY                1033    //"New Order
Missing Qty key "Qty##*".
#define ERR_NEWORDER_QTY_INVALID                    1034    //"New Order
Qty invalid must be numeric range 1 - 99.
#define ERR_NEWORDER_SUPPW_RANGE                    1035    //"New Order
Supp_W value out of range range = 1 - Max Warehouses.
#define ERR_NEWORDER_ITEMID_RANGE                   1036    //"New Order
Item Id is out of range. Range = 1 to 999999.
#define ERR_NEWORDER_QTY_RANGE                              1037
        //"New Order Qty is out of range. Range = 1 to 99.
#define ERR_PAYMENT_DISTRICT_INVALID                1038    //"Payment
District ID is invalid must be 1 - 10.
#define ERR_NEWORDER_SUPPW_WITHOUT_ITEMID       1039    //"New Order Supp_W
field entered without a corrisponding Item_Id.
#define ERR_NEWORDER_QTY_WITHOUT_ITEMID             1040    //"New Order
Qty entered without a corrisponding Item_Id.
#define ERR_NEWORDER_NOITEMS_ENTERED                1041    //"New Order
Blank Items between items, items must be continuous.
#define ERR_PAYMENT_MISSING_DID_KEY                         1042
        //"Payment missing District Key "DID*".
#define ERR_PAYMENT_DISTRICT_RANGE                          1043
        //"Payment District Out of range, range = 1 - 10.
#define ERR_PAYMENT_MISSING_CID_KEY                         1044
        //"Payment missing Customer Key "CID*".
#define ERR_PAYMENT_CUSTOMER_INVALID                1045    //"Payment
Customer data type invalid, must be numeric.
#define ERR_PAYMENT_MISSING_CLT                             1046
        //"Payment missing Customer Last Name Key "CLT*".
#define ERR_PAYMENT_LAST_NAME_TO_LONG               1047    //"Payment
Customer last name longer than 16 characters.
#define ERR_PAYMENT_CUSTOMER_RANGE                          1048
        //"Payment Customer ID out of range, must be 1 to 3000.
#define ERR_PAYMENT_CID_AND_CLT                             1049
        //"Payment Customer ID and Last Name entered must be one or other.
#define ERR_PAYMENT_MISSING_CDI_KEY                         1050
        //"Payment missing Customer district key "CDI*".
#define ERR_PAYMENT_CDI_INVALID                             1051
        //"Payment Customer district invalid must be numeric.
#define ERR_PAYMENT_CDI_RANGE                               1052
        //"Payment Customer district out of range must be 1 - 10.
#define ERR_PAYMENT_MISSING_CWI_KEY                         1053
        //"Payment missing Customer Warehouse key "CWI*".
#define ERR_PAYMENT_CWI_INVALID                             1054
        //"Payment Customer Warehouse invalid must be numeric.
#define ERR_PAYMENT_CWI_RANGE                               1055
        //"Payment Customer Warehouse out of range, 1 to Max Warehouses.
```

```
#define ERR_PAYMENT_MISSING_HAM_KEY                                 1056
        //"Payment missing Amount key "HAM*".
#define ERR_PAYMENT_HAM_INVALID                                     1057
        //"Payment Amount invalid data type must be numeric.
#define ERR_PAYMENT_HAM_RANGE                                       1058
        //"Payment Amount out of range, 0 - 9999.99.
#define ERR_ORDERSTATUS_MISSING_DID_KEY                   1059      //"Order
Status missing District key "DID*".
#define ERR_ORDERSTATUS_DID_INVALID                                 1060
        //"Order Status District invalid, value must be numeric 1 - 10.
#define ERR_ORDERSTATUS_DID_RANGE                         1061      //"Order
Status District out of range must be 1 - 10.
#define ERR_ORDERSTATUS_MISSING_CID_KEY                   1062      //"Order
Status missing Customer key "CID*".
#define ERR_ORDERSTATUS_MISSING_CLT_KEY                   1063      //"Order
Status missing Customer Last Name key "CLT*".
#define ERR_ORDERSTATUS_CLT_RANGE                         1064      //"Order
Status Customer last name longer than 16 characters.
#define ERR_ORDERSTATUS_CID_INVALID                                 1065
        //"Order Status Customer ID invalid, range must be numeric 1 - 3000.
#define ERR_ORDERSTATUS_CID_RANGE                         1066      //"Order
Status Customer ID out of range must be 1 - 3000.
#define ERR_ORDERSTATUS_CID_AND_CLT                                 1067
        //"Order Status Customer ID and LastName entered must be only one."
#define ERR_DELIVERY_MISSING_OCD_KEY                      1068      //"Delivery
missing Carrier ID key \"OCD*\".
#define ERR_DELIVERY_CARRIER_INVALID                      1069      //"Delivery
Carrier ID invalid must be numeric 1 - 10.
#define ERR_DELIVERY_CARRIER_ID_RANGE                     1070      //"Delivery
Carrier ID out of range must be 1 - 10.
#define ERR_PAYMENT_MISSING_CLT_KEY                                 1071
        //"Payment missing Customer Last Name key "CLT*".

#endif
```

File: errorstring.c

```
#include <windows.h>
#include "errorstring.h"

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

        LPVOID lpMsgBuf;

        int nBytes;

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

        if (!nBytes)
        {
                snprintf(buf, buf_size, "Unable to get error message for error
%d", dwError);
        }
}
```

File: errorstring.h

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

File: 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
```

File: 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
```

File: 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_
```

File: 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"

HICON           hIcon;
HINSTANCE       hInst;

DWORD                   versionExeMS;
DWORD                   versionExeLS;
DWORD                   versionExeMM;
DWORD                   versionDllMS;
DWORD                   versionDllLS;

static  BOOL    bLog;
static  BOOL    bConnectionPooling;
static  int             iThreads;
static  int             iMaxWareHouse;
static  int             iDelayMs;
static  int             iDeadlockRetry;
static  int             iMaxConnections;
static  int             iPoolThreadLimit;
static  int             iThreadTimeout;
static  int             iListenBackLog;
static  int             iAcceptExOutstanding;
static  int             iDllType;

static  int             iMaxPhysicalMemory;             //max physical
memory in MB
```

```
static  int             iConnectDelay;
static  char    szVersion[256];

BOOL    CALLBACK UpdatedDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
BOOL    CALLBACK MainDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
BOOL    CALLBACK CopyDlgProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM
lParam);
static  void            ProcessOK(HWND hwnd, char *szDllPath);
static  void            ReadRegistrySettings(void);
static  void            WriteRegistrySettings(char *szDllPath);
static  int                     CopyFiles(HWND hDlg, char *szDllPath);
static  BOOL            GetInstallPath(char *szDllPath);
static  void            GetVersionInfo(char *szDLLPath, char *szExePath);
static  BOOL            CheckWWWebService(void);
static  BOOL            StartWWWebService(void);
static  BOOL            StopWWWebService(void);
static  void            UpdateDialog(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)
{
        PAINTSTRUCT             ps;
        MEMORYSTATUS    memoryStatus;
        char                    szTmp[256];
        static char             szDllPath[256];
        static char             szExePath[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)
{
        int             d;
        HWND    hDlg;
        int             rc;

        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)
{
        HKEY    hKey;
        DWORD   size;
        DWORD   type;
        char    szTmp[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)
{
        HKEY    hKey;
        DWORD   dwDisposition;
        char    szTmp[256];
        char    *ptr;
        int             iRc;

        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)
```

```
{
        HGLOBAL                 hDLL;
        HGLOBAL                 hExe;
        HRSRC                   hResInfo;
        BYTE                    *pSrc;
        HANDLE                  hFile;
        DWORD                   dwSize;
        DWORD                   d;
        char                    szTmp[256];
        char                    *ptr;
        BOOL                    bSvcRunning;


        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)
{
    HKEY hKey;
        BYTE    szTmp[256];
        char    szKey[256];
        DWORD   size;
        DWORD   sv;
        BOOL    bRc;
        int             len;
        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)
{
        DWORD                           d;
        DWORD                           dwSize;
        DWORD                           dwBytes;
        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, &dwBytes);
```

```
                versionExeMS = vs->dwProductVersionMS;
                versionExeLS = LOWORD(vs->dwProductVersionLS);
                versionExeMM = HIWORD(vs->dwProductVersionLS);
                free(ptr);
        }
        return;
}

static BOOL CheckWWWebService(void)
{
        SC_HANDLE               schSCManager;
        SC_HANDLE               schService;
        SERVICE_STATUS   ssStatus;

        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_HANDLE               schSCManager;
        SC_HANDLE               schService;
        SERVICE_STATUS   ssStatus;
        DWORD                   dwOldCheckPoint;

        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_HANDLE               schSCManager;
        SC_HANDLE               schService;
        SERVICE_STATUS   ssStatus;
        DWORD                   dwOldCheckPoint;

        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;
}
```

File: 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_SIZE 1017
#define ED_IIS_THREAD_TIMEOUT           1018
```

```
#define ED_IIS_LISTEN_BACKLOG            1019
#define IDC_DBLIB                        1021
#define IDC_ODBC                         1022
#define IDC_CONNECT_POOL                                 1023
#define ED_USER_CONNECT_DELAY_TIME                       1024

// 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
```

File: 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 c 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
```

File: pipe_routines.c

```
#include <windows.h>
#include <stdio.h>
#include "pipe_routines.h"
#include "trans.h"
#include "tpcc.h"
#include "tux.h"

const int MAXRETRIES=600;
const int SLEEP_TIME=100;

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_ATTRIBUTES             sa;
        PSECURITY_DESCRIPTOR     pSD;

        _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;
        int nRetries;

        _snprintf(PipeName, sizeof(PipeName), CLIENT_PIPE_PATH, ClientNumber);
#ifdef _DEBUG
        fprintf(stderr, "OpenClientPipe begins for client %d (%s)\n",
ClientNumber, PipeName);
#endif

        for(nRetries=0;nRetries<MAXRETRIES;nRetries++)
        {
                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(SLEEP_TIME);
                                break;
                        default:
                                fprintf(stderr, "OpenClientPipe(%d): error in
create of %s.  Error = %d\n", ClientNumber, PipeName, GetLastError());
                                return INVALID_HANDLE_VALUE;
                                break;
                }
        }

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

                return INVALID_HANDLE_VALUE;
        }

        if (!SetNamedPipeHandleState(hPipe, &DesiredMode, NULL, NULL))
        {
                fprintf(stderr, "OpenClientPipe(%d), 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;
}
```

File: 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
```

File: 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
```

File: 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);

/* #define EXTRA_DEBUG /* */

static  CRITICAL_SECTION         ErrorLogCriticalSection;

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               *dbproc                 DBPROCESS id
pointer
 *                              int                             severity
        severity of error
 *                              int                             dberr
                error id
 *                              int                             oserr
                operating system specific error code
```

```
 *                              char                    *dberrstr
        printable error description of dberr
 *                              char                    *oserrstr
        printable error description of oserr
 *
 * RETURNS:             int                             INT_CONTINUE
        continue 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)
        {
                PECBINFO                        pEcbInfo;
                EXTENSION_CONTROL_BLOCK  *pECB;
                FILE                                    *fp;
                SYSTEMTIME                              systemTime;
                char                                    szTmp[256];
                int                                             iTermId;
                int                                             iSyncId;
                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               *dbproc                 DBPROCESS id
pointer
 *                              DBINT                   msgno
        message number
```

```
 *                              int                             msgstate
        message state
 *                              int                             severity
        message severity
 *                              char                    *msgtext
        printable message description
 *
 * RETURNS:             int                             INT_CONTINUE
        continue if error is SQLETIME else INT_CANCEL action
 *                                                              INT_CANCEL
                cancel operation
 *
 * COMMENTS:    This function also sets the dead lock dbproc variable if
necessary.
 *
 */

int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int severity,
char *msgtext)
{
        PECBINFO                        pEcbInfo;
        EXTENSION_CONTROL_BLOCK  *pECB;
        FILE                                    *fp;
        SYSTEMTIME                              systemTime;
        char                                    szTmp[256];
        int                                             iTermId;
        int                                             iSyncId;

        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  *pECB    passed in structure pointer
from inetsrv.
 *                               int                             iTermId
        terminal id of browser
 *                               int                             iSyncId
        sync id of browser
 *                               DBPROCESS               **dbproc pointer to
returned DBPROCESS
 *                               char                    *server         SQL
server name
 *                               char                    *database       SQL
server database
 *                               char                    *user           user
name
 *                               char                    *password       user
password
 *                               char                    *app
        pointer to returned application array
 *                               int                             *spid
        pointer to returned spid
 *                               long                    *pack_size
        pointer to returned default pack size
 *
 * RETURNS:              BOOL    FALSE   if successfull
 *                                               TRUE    if 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)
        {

                RETCODE         rc;
                char            buffer[30];
                PECBINFO pEcbInfo;

                *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  *pECB    passed in structure pointer
from inetsrv.
 *                              DBPROCESS               *dbproc pointer to
DBPROCESS
 *
 * RETURNS:              BOOL    FALSE   if successfull
 *                                       TRUE    if 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  *pECB                    passed in
structure pointer from inetsrv.
 *                                  int
        iTermId                 terminal id of browser
 *                                  int
        iSyncId                 sync id of browser
 *                                  DBPROCESS
        *dbproc                 connection db process id
 *                                  STOCK_LEVEL_DATA        *pStockLevel
        stock level input / output data structure
 *                                  short
        deadlock_retry  retry count if deadlocked
 *
 * RETURNS:              BOOL    FALSE           if successfull
 *                                               TRUE            if
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)
        {
                int                     tryit;
                PECBINFO pEcbInfo;


                //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)
        {
                int                     tryit;
                RETCODE         rc;
                char            printbuf[25];
                BYTE            *pData;
                PECBINFO pEcbInfo;

                //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  *pECB                    passed in
structure pointer from inetsrv.
 *                                       int
        iTermId                         terminal id of browser
 *                                       int
        iSyncId                         sync id of browser
 *                                       DBPROCESS
        *dbproc                         connection db process id
 *                                       NEW_ORDER_DATA                    *pNewOrder
                pointer to new order structure for input/output data
 *                                       short
        deadlock_retry   retry count if deadlocked
 *
 * RETURNS:              int     TRUE    transaction committed
 *                                       FALSE   item number not valid
 *                                       -1              deadlock 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)
```

```
{
        int                     i;
        int                     j;
        int                     tryit;
        DBINT           commit_flag;
        char            buffer[255];
        PECBINFO pEcbInfo;

        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)
        {
                RETCODE                 rc;
                int                             i;
                DBINT                   commit_flag;
                int                             tryit;
                char                    printbuf[25];
                char                    tmpbuf[30];
                DBDATETIME              datetime;
                BYTE                    *pData;
                PECBINFO        pEcbInfo;
#ifdef EXTRA_DEBUG
                char ExtraDebugBuf[8192]={0};

                sprintf(ExtraDebugBuf, "new_order begins w_id = %d, d_id=%d,
c_id=%d o_ol_cnt=%d\n",
                                        pNewOrder->w_id,
```

```
                                        pNewOrder->d_id,
                                        pNewOrder->c_id,
                                        pNewOrder->o_ol_cnt);
#endif
                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++)
                {
#ifdef EXTRA_DEBUG
                        if (tryit)
                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf),
"\tretry %d\n");
#endif
                        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++)
                                {
#ifdef EXTRA_DEBUG

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "\ti=%d ol_i_id=%d,
ol_supply_w_id=%d, ol_quantity=%d\n", i, pNewOrder->Ol[i].ol_i_id, pNewOrder-
>Ol[i].ol_supply_w_id,  pNewOrder->Ol[i].ol_quantity);
#endif
                                        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)
                                {
#ifdef EXTRA_DEBUG

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "\tdbrpcexec succeeded,
retstatus=%d\n", dbretstatus(dbproc));
#endif
                                        pNewOrder->total_amount=0;

                                        // Get resutls from order line
                                        for (i = 0; i<pNewOrder->o_ol_cnt;
i++)
                                        {
#ifdef EXTRA_DEBUG

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "\tresults line %d ", i);
#endif
                                                if (((rc = dbresults(dbproc))
!= NO_MORE_RESULTS) && (rc != FAIL))
                                                {
#ifdef EXTRA_DEBUG

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "dbnumcols=%d\n",
dbnumcols(dbproc));
#endif
                                                        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;
                                                        }
                                                }
                                        }
#ifdef EXTRA_DEBUG
                                        else
                                                if (rc !=
NO_MORE_RESULTS)
        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "for loop dbresults
returned rc=%d\n", rc);
#endif
                                }
#ifdef EXTRA_DEBUG
        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "\tstarting while");
#endif
                                        while (((rc = dbresults(dbproc)) !=
NO_MORE_RESULTS) && (rc != FAIL))
                                        {
#ifdef EXTRA_DEBUG
        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf)," dbnumcols=%d",
dbnumcols(dbproc));
#endif
                                                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);
                                                        }
                                                }
                                }
#ifdef EXTRA_DEBUG

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), " second while ends, rc=%d
commit_flag = %d\n", rc, commit_flag);
#endif
                                }
#ifdef EXTRA_DEBUG
                                else

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "\tdbrpcexe !=
SUCCESS\n");
#endif
                        }
#ifdef EXTRA_DEBUG
                        else
                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf),
"dbrpcinit failed\n");
#endif
                        if (SQLDetectDeadlock(dbproc))
                        {
                                pNewOrder->num_deadlocks++;
                                sprintf(printbuf,"deadlock: retry:
%d",pNewOrder->num_deadlocks);
                                Sleep(DEADLOCKWAIT*tryit);
                        }
                        else
                        {
#ifdef EXTRA_DEBUG
                                FILE *fp = fopen(szErrorLogPath, "ab");


        EnterCriticalSection(&ErrorLogCriticalSection);
                                fputs(ExtraDebugBuf, fp);

        LeaveCriticalSection(&ErrorLogCriticalSection);

                                fclose(fp);
#endif
                                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.  ");
#ifdef EXTRA_DEBUG
                {
                        FILE *fp = fopen(szErrorLogPath, "ab");


        EnterCriticalSection(&ErrorLogCriticalSection);
```

```
                                fputs(ExtraDebugBuf, fp);

        LeaveCriticalSection(&ErrorLogCriticalSection);

                                fclose(fp);
                }
#endif
                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  *pECB                  passed in
structure pointer from inetsrv.
 *                              int
        iTermId                 terminal id of browser
 *                              int
        iSyncId                 sync id of browser
 *                              DBPROCESS
        *dbproc                 connection db process id
 *                              PAYMENT_DATA                    *pPayment
                pointer to payment input/output data structure
 *                              short
        deadlock_retry   deadlock retry count
 *
 * RETURNS:              int     TRUE            success
 *                                       -1                      max
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)
        {
                int                     tryit;
                char            printbuf[25];
                char            buffer[255];
                BOOL            deadlock_detected;

                PECBINFO pEcbInfo;

                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)
        {
                RETCODE         rc;
                int                     tryit;
                char            printbuf[26];
                DBDATETIME      datetime;
                BYTE            *pData;
                PECBINFO pEcbInfo;
#ifdef EXTRA_DEBUG
                PAYMENT_DATA orig_data;
                char ExtraDebugBuf[2048]={0};

                memcpy(&orig_data, pPayment, sizeof(orig_data));
                sprintf(ExtraDebugBuf, "payment begins w_id = %d, c_w_id=%d,
h_amout=%.2lf, d_id=%d, c_d_id=%d, c_id=%d c_last=%s\n",
                                        orig_data.w_id,
                                        orig_data.c_w_id,
                                        orig_data.h_amount,
                                        orig_data.d_id,
                                        orig_data.c_d_id,
                                        orig_data.c_id,
                                        orig_data.c_id?"":orig_data.c_last);
#endif

                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);
                                }
                        }
#ifdef EXTRA_DEBUG
                        else
```

```
                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf),
"\tdbrpcinit failed\n");
#endif
                        if (dbrpcexec(dbproc) == SUCCEED)
                        {
#ifdef EXTRA_DEBUG
                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf),
"\tdbrpcexec == SUCCEED");
#endif
                                while (((rc = dbresults(dbproc)) !=
NO_MORE_RESULTS) && (rc != FAIL))
                                {
                                        if (DBROWS(dbproc) &&
(dbnumcols(dbproc) == 27))
                                        {
#ifdef EXTRA_DEBUG

        sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), " dbnumcols=%d",
dbnumcols(dbproc));
#endif
                                                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));
                                                                }
                                                        }
                                                }
#ifdef EXTRA_DEBUG
                                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf), "
while ends, rc=%d (%s), status=%d\n", rc,
rc==NO_MORE_ROWS?"NO_MORE_ROWS":(rc==FAIL?"FAIL":"UNKNOWN"),
                                                        dbretstatus(dbproc));
#endif
                                }
                                else
                                {
#ifdef EXTRA_DEBUG

                                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf),
"\tdbrpcexed != SUCCEED\n");
                                                sprintf(ExtraDebugBuf+strlen(ExtraDebugBuf),
                                                        "\tw_id = %d, c_w_id=%d,
h_amount=%.2lf, d_id=%d, c_d_id=%d, c_id=%d\n",
                                                                pPayment->w_id,
                                                                pPayment->c_w_id,
                                                                pPayment->h_amount,
                                                                pPayment->d_id,
                                                                pPayment->c_d_id,
                                                                pPayment->c_id);

                                                sprintf(pPayment->execution_status, "dbrpcexec
!= SUCCEED, w_id = %d, c_w_id=%d, h_amount=%lf, d_id=%d, c_d_id=%d, c_id=%d\n",
                                                                orig_data.w_id,
                                                                orig_data.c_w_id,
                                                                orig_data.h_amount,
                                                                orig_data.d_id,
                                                                orig_data.c_d_id,
                                                                orig_data.c_id);
                                                sprintf(pPayment-
>execution_status+strlen(pPayment->execution_status),
                                                                "                now:w_id = %d,
c_w_id=%d, h_amount=%lf, d_id=%d, c_d_id=%d, c_id=%d\n",
                                                                pPayment->w_id,
```

```
                                        pPayment->c_w_id,
                                        pPayment->h_amount,
                                        pPayment->d_id,
                                        pPayment->c_d_id,
                                        pPayment->c_id);
                                {
                                        FILE *fp = fopen(szErrorLogPath,
"ab");

        EnterCriticalSection(&ErrorLogCriticalSection);
                                        fprintf(fp, "%s\n", pPayment-
>execution_status);

        LeaveCriticalSection(&ErrorLogCriticalSection);

                                        fclose(fp);
                                }
#else
                                strcpy(pPayment-
>execution_status,"dbrpcexec(dbproc) != SUCCEED");
#endif
                        }

                        if (SQLDetectDeadlock(dbproc))
                        {
                                pPayment->num_deadlocks++;
                                sprintf(printbuf,"deadlock: retry:
%d",pPayment->num_deadlocks);
                                Sleep(DEADLOCKWAIT*tryit);
                        }
                        else
                        {
#ifdef EXTRA_DEBUG
                                FILE *fp = fopen(szErrorLogPath, "ab");

        EnterCriticalSection(&ErrorLogCriticalSection);
                                fputs(ExtraDebugBuf, fp);

        LeaveCriticalSection(&ErrorLogCriticalSection);
                                fclose(fp);
#endif
                                if ( pPayment->c_id == 0 )
                                {
                                        strcpy(pPayment-
>execution_status,"Invalid Customer id,name.");
                                        return 0;
                                }
                                else
                                        strcpy(pPayment-
>execution_status,"Transaction commited.");
                                        return TRUE;
                                }
                        }

#ifdef EXTRA_DEBUG
                        {
                                FILE *fp = fopen(szErrorLogPath, "ab");
                                EnterCriticalSection(&ErrorLogCriticalSection);
                                fputs(ExtraDebugBuf, fp);
                                LeaveCriticalSection(&ErrorLogCriticalSection);
                                fclose(fp);
                        }
#endif
                        // 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  *pECB                    passed in
structure pointer from inetsrv.
 *                                      int
        iTermId                 terminal id of browser
 *                                      int
        iSyncId                 sync id of browser
 *                                      DBPROCESS
        *dbproc                 connection db process id
 *                                      ORDER_STATUS_DATA
        *pOrderStatus   pointer to Order Status data input/output structure
 *                                      short
        deadlock_retry  deadlock retry count
 *
```

```
 * RETURNS:              int      -1              max deadlock reached
 *                                       0               No orders found for
customer
 *                                       1               Transaction
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];
                PECBINFO pEcbInfo;

                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)
	{
		RETCODE		rc;
		int			tryit;
		int			i;
		char		printbuf[25];
		DBDATETIME	datetime;
		BYTE		*pData;
		PECBINFO pEcbInfo;

		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                       *dbproc
        connection db process id to check
 *
 * RETURNS:             BOOL    FALSE           no deadlock detected
 *                                              TRUE            deadlock
condition exists
 *
 * COMMENTS:    None
 *
 */

BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
{
        PECBINFO pEcbInfo;

        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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *                               void            *uPtr   returned
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:    DBRPOCESS       dbproc  ODBC 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:    PDBPROCESS      dbproc          pointer to odbc
dbprocess structure
         *                               UWORD   ipar            Parameter
number, ordered sequentially left to right, starting at 1.
         *                               SWORD   fParamType      The type of
the parameter.
         *                               SWORD   fCType          The C data
type of the parameter.
         *                               SWORD   fSqlType The SQL data type of
the parameter.
         *                               UDWORD  cbColDef The precision of the
column or expression
         *
        of the corresponding parameter marker.
         *                               SWORD   ibScale         The scale of
the column or expression of the corresponding
         *
        parameter marker.
         *                               PTR             rgbValue A pointer to
a buffer for the parameter s data.
```

```
         *                               SDWORD   cbValueMax       Maximum
length of the rgbValue buffer.
         *                               void             *uPtr    returned
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:    DBRPOCESS       dbproc   ODBC dbprocess structure
         *
         * RETURNS:              none
         *
         * COMMENTS:     none
         *
         */

        void ODBCError(PDBPROCESS dbproc)
        {
                SDWORD          lNativeError;
                char            szState[6];
                char            szMsg[SQL_MAX_MESSAGE_LENGTH];
                char            szMsgText[256];
                PECBINFO pEcbInfo;
                char            szTmp[256];
                FILE            *fp;
                SYSTEMTIME      systemTime;

                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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *                               char            *szStatement    sql
stored procedure statement to be executed.
         *
         * RETURNS:              none
         *
         * COMMENTS:     none
         *
         */

        BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement)
        {
                RETCODE         rc;
                PECBINFO pEcbInfo;

                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:    DBRPOCESS       dbproc  ODBC dbprocess structure
         *                               UWORD   icol
        Column number of result data, ordered sequentially left to right,
starting at 1.
         *                               SWORD   fCType                  The
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
         *                               PTR             rgbValue
        Pointer to storage for the data. If rgbValue is a null pointer, the
         *
                driver unbinds the column.
         *                               SDWORD  cbValueMax
        Maximum 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)
        {
                RETCODE         rc;
                PECBINFO pEcbInfo;

                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:    DBRPOCESS        dbproc   ODBC 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:    DBRPOCESS        dbproc   ODBC dbprocess structure
         *
         * RETURNS:              none
         *
         * COMMENTS:     none
         *
         */

        BOOL MoreResults(PDBPROCESS  dbproc)
        {
                PECBINFO pEcbInfo;

                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:    DBRPOCESS        dbproc   ODBC dbprocess structure
         *
         * RETURNS:              FALSE if successfull
         *                               TRUE if an error occurs
         *
         * COMMENTS:     none
         *
         */

        BOOL ReopenConnection(PDBPROCESS dbproc)
        {
                RETCODE          rc;
                PECBINFO pEcbInfo;
                int                      iCount;
                FILE             *fp;
                SYSTEMTIME       systemTime;
```

```
                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);
}
```

File: sqlroutines.h

```
#ifndef PERF_MEASURE
#define PERF_MEASURE
#endif

//this structure allows the EXTENSION CONTROL BLOCK to be passed to the msg and
error handlers.
typedef struct _ECBINFO
{
        int                                             iTermId; //terminal
id
        int                                             iSyncId; //browser
sync id
#ifdef PERF_MEASURE
        int                                             iTMms;
        int                                             iSUTms;
#endif
        BOOL                                    bDeadlock;      //deadlock
condition flag
        BOOL                                    bFailed; //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);
```

File: tpcc.c

```
/*      FILE:           TPCC.C
 *                              Microsoft TPC-C Kit Ver. 3.00.000
 *                              Audited 08/23/96 By 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
        HENV    henv;
#endif
```

```
char    szServer[32]     = { 0 };         //global variables used with this DLL
char    szUser[32]               = { 0 };
char    szPassword[32]   = { 0 };
char    szDatabase[32]   = "tpcc";
BOOL    bLog                     = FALSE;
int             iThreads         = 5;
int             iMaxWareHouses   = 500;
int             iQSlotts         = 3000;
int             iDelayMs         = 100;
int             iConnectDelay    = 500;
short   iDeadlockRetry   = (short)3;
short   iMaxConnections = (short)25;

#ifdef USE_ODBC
        int             bConnectionPooling = 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
TERM    Term = { 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 char     szTpccLogPath[256];     //path to html log file if logging
turned on in registry.
char szErrorLogPath[256];//path to error log file.

static  CRITICAL_SECTION        CriticalSection;

static  LPTSTR                                  lpszPipeName    =
TEXT("\\\\.\\pipe\\DELISRV");
static  HANDLE                                  hDeliveryWrite  =
INVALID_HANDLE_VALUE;
static  HANDLE                                  hPipe                   =
INVALID_HANDLE_VALUE;
```

```
EXTENSION_CONTROL_BLOCK  *gpECB;
static  int                                             bTpccExit;
        //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:    HANDLE  hModule                           module handle
 *                               DWORD   ul_reason_for_call       reason for
call
 *                               LPVOID  lpReserved
        reserved for future use
 *
 * RETURNS:              BOOL    FALSE                             errors
occured in initialization
 *                                              TRUE
        DLL successfully initialized
 *
 * COMMENTS:     None
 *
 */

BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID
lpReserved)
{
        int                             i;
        static SECURITY_ATTRIBUTES      sa;
        static PSECURITY_DESCRIPTOR pSD;

        switch( ul_reason_for_call )
        {
                case DLL_PROCESS_ATTACH:
                        {
                                const char
*LogFile="\\temp\\tpcc_logs\\tpcc.dll.txt";
                                mkpath(LogFile);
                                freopen(LogFile, "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    *ptr    void 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)
{
        int                                             l, d;
        SECURITY_ATTRIBUTES             sa;
        PSECURITY_DESCRIPTOR    pSD;
```

```
        // 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 *pVer    passed in structure in which to place
expected version number.
 *
 * RETURNS:              TRUE     inet 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  *pECB    structure pointer to passed
in internet
 *
                        service information.
 *
 * RETURNS:              DWORD    HSE_STATUS_SUCCESS
                connection can be dropped if error
 *
        HSE_STATUS_SUCCESS_AND_KEEP_CONN  keep connect valid comment sent
 *
 * COMMENTS:     None
 *
 */

static int
ExceptionFilter(int iException, int iLine, char *pFile)
{
        Log("thread %d caught exception 0x%x from line %d in file %s\n",
                GetCurrentThreadId(), iException, iLine, pFile);

        return EXCEPTION_EXECUTE_HANDLER;
}

DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
        int iCmd, FormId, TermId, iSyncId;
        FILE *fp;
#ifdef TRANS_SEQ
```

```
        int seq;
#endif

        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 )
        {
                SYSTEMTIME      systemTime;

                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) )
                {
                        fprintf(stderr, "Invalid TermID %d\n");
                        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 )
                {
                        fprintf(stderr, "Bad TermId SyncId pair - Termid=%d
SyncId=%d Actual SyncID=%d w_id=%d d_id=%d TickCount=%u\n",
                                        TermId, iSyncId,
Term.pClientData[TermId].iSyncId, Term.pClientData[TermId].w_id,
Term.pClientData[TermId].d_id, Term.pClientData[TermId].iTickCount);
                        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();

        __try
        {
#ifdef TRANS_SEQ
                {
                        char *p = strstr(pECB->lpszQueryString, "SEQ=");

                        seq = -1;

                        if (p)
                        {
                                seq=strtoul(p+4, NULL, 0);

                                Log("B 0x%x\n", seq);
                        }

                }
#endif

                //go execute http: command
```

```
                (*DoCmd[iCmd])(pECB, FormId, TermId, iSyncId);

#ifdef TRANS_SEQ
                if (seq != -1)
                        Log("E 0x%x\n", seq);
#endif
        }
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                fprintf(stderr, "caught an exception in HttpExtensionProc\n");
        }
        //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:    int                             TermId
        client terminal id
 *
 * RETURNS:              BOOL    FALSE
        Terminal ID Invalid
 *                                              TRUE
                        Terminal 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  *pECB           structure pointer to
passed in internet
 *
                                service information.
 *                              int                             *pCmd
                        returned command id
 *                              int                             *pFormId
                returned active form client browser is on
 *                              int                             *pTermId
                returned client terminal id
 *
 * RETURNS:              BOOL    FALSE
        success
 *                                                      TRUE
                        command 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:   int                                                iFormId
        unused
 *                              int
        iTermId         id of calling browser, i.e. TERMID= from http command
line
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                              iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                              iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                                  iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling borwser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                                 iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                                 iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                                 iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling browser
```

```
 *                                EXTENSION_CONTROL_BLOCK  *pECB
        structure 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)
{
        int     iCurrent;

        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:    int                                             iFormId
        unused
 *                              int
        iTermId         id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId         sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                             iFormId
        unused
 *                              int
        iTermId         id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId         sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure pointer to passed in internet
 *
                                service information.
 * RETURNS:             None
 *
 * COMMENTS:     None
 *
 */


void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int
iSyncId)
{
#ifdef BOIL_TEST
        int iMod = Term.pClientData[iTermId].w_id % 15;

        Sleep((iMod?iMod:25)*1000);

        WriteZString(pECB, MakeMainMenuForm(iTermId,
Term.pClientData[iTermId].iSyncId) );
        return;
#endif
        __try
        {
        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;
        }
        }
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                fprintf(stderr, "caught an exception in ProcessCmd\n");
        }

}

/* FUNCTION: void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
 *
 * PURPOSE:      This function frees all currently logged in terminal ids.
 *
```

```
 * ARGUMENTS:    int                                                 iFormId
        unused
 *                              int
        iTermId         id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId         sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                                 iFormId
        unused
 *                              int
        iTermId         id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId         sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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:    int                                                iFormId
        unused
 *                              int
        iTermId          id of calling browser, i.e. TERMID= from http command
line
 *                              int
        iSyncId          sync id of calling browser
 *                              EXTENSION_CONTROL_BLOCK  *pECB
        structure 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;
        int     iTotal;

        // 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  *pECB   passed in structure pointer
from inetsrv.
 *                              char                    *szStr
        string 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;
        int     lpbSize;
        int     iSize;
        char    szHeader[128];
        char    szHeader1[128];

        lpbSize = strlen(szStr)+1;

        if ( bLog )
        {
                SYSTEMTIME      systemTime;

                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  *pECB    passed in structure pointer
from inetsrv.
 *                               char                    *format
        printf 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                    *pQueryString    http string from
client browser
 *                               char                    *pKey
        key value to look for
 *                               char                    *pValue
        character array into which to place key's value
 *                               int                             iMax
                maximum length of key value array.
 *
 * RETURNS:              BOOL    FALSE   key value not found
 *                                               TRUE    key 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:              int     TRUE or 1 if sucessfull
 *                               int     FALSE or 0 if terminal id cannot be
allocated.
 *
 * COMMENTS:     None
 *
 */

static int TermAllocate(void)
{
        Term.iAvailable =  iMaxConnections;

        if ( Term.pClientData )
        {
                fprintf(stderr, "trying to realloc in TermAllocate()\n");
                return 0;
        }

        Term.pClientData = (PCLIENTDATA)malloc(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  *pECB                    passed in
structure pointer from inetsrv.
 *                              char
        *pQueryString    http query string passed to this DLL.
 *
 * RETURNS:              int             assigned terminal id
 *                                       -1      cannot 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)
{
        char    szTmp[32];
        int             i, iCurrent;
        static int iHint=0;

        EnterCriticalSection(&CriticalSection);

        if (iHint < Term.iAvailable && !Term.pClientData[iHint].inUse)
                iCurrent = iHint;
        else
        {
                int iTotalConnections, iTickCount;
                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;
                                }
                        }
                        fprintf(stderr, "iTotalConnections(%d) >=
iMaxConnections(%d), stealing entry %d\n",
                                iTotalConnections, iMaxConnections, iCurrent);
                }
                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 TermAddErr2;
                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;
        }

        if ((iCurrent % 100) == 0 ||
            Term.pClientData[iCurrent].iSyncId != iCurrent)
                fprintf(stderr, "TermAdd: TermID %d (w_id %d, d_id %d) gets
Syncid %d, TickCount=%u\n",
                        iCurrent, Term.pClientData[iCurrent].w_id,
                        Term.pClientData[iCurrent].d_id,
                        Term.pClientData[iCurrent].iSyncId,
                        Term.pClientData[iCurrent].iTickCount);

        iHint = iCurrent+1;

        LeaveCriticalSection(&CriticalSection);
        return iCurrent;

TermAddErr1:
        Term.pClientData[iCurrent].inUse = 0;
TermAddErr2:
        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  *pECB   passed in structure pointer
from inetsrv.
 *                              int
        id              Terminal 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  *pECB           passed in structure
pointer from inetsrv.
 *                              int
        iTermId         id of browser client that this connection is for.
 *                              int
        iSyncId         sync id for this client session
 *                              char
        *szServer       sql server name
 *                              char
        *szUser         user name
 *                              char
        *szPassword     user password
 *                              char
        *szDatabase     database to use
 *
 * RETURNS:             BOOL    FALSE   if successfull
 *                                              TRUE    if 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)
{
        char    szApp[32];
        char    server[256];
        char    database[256];
```

```
        char    user[256];
        char    password[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  *pECB    passed in structure pointer
from inetsrv.
 *                              int
        iTermId id of browser client that this connection is for.
 *                              int
        iSyncId sync id of client browser
 *
 * RETURNS:              BOOL    FALSE   if successfull
 *                                               TRUE    if an error occurs
and connection cannot be terminated.
 *
 * COMMENTS:     None
 *
 */

static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
        PECBINFO pEcbInfo;

        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            *szDest Destination buffer where formatted
string is to be placed
 *                                char            *szPic  picture string which
describes how character value is to be
 *
        formatted.
 *                                char            *szSrc  character 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:    int                             iTermId client browser
terminal id
 *                               int                             iSyncId
        client browser sync id
 *                               BOOL                    bInput   TRUE 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=\"\">");
        else
        {
#ifdef PERF_MEASURE
                {
                        PECBINFO pEcbInfo =
SQLGetECB(Term.pClientData[iTermId].dbproc);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T6\" VALUE=\"%d\">", pEcbInfo->iTMms);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T7\" VALUE=\"%d\">", pEcbInfo->iSUTms);
                }
#endif
        }
        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:    int                             iTermId client browser
terminal id
 *                              int                             iSyncId
        client 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);
#ifdef BOIL_TEST
        strcat(szForm,  "Order Number: 6<BR>");
#endif
        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:    int                             iTermId client browser
terminal id
 *                               int                               iSyncId
        client browser sync id
 *                               BOOL                    bInput   TRUE if form
is being constructed for input else FALSE
 *                               BOOL                    bValid   TRUE 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;
        char    szName[146];
        char    szCredit[14];
        int             i;

        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);
#ifdef PERF_MEASURE
                {
                        PECBINFOpEcbInfo =
SQLGetECB(Term.pClientData[iTermId].dbproc);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T6\" VALUE=\"%d\">", pEcbInfo->iTMms);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T7\" VALUE=\"%d\">", pEcbInfo->iSUTms);
                }
#endif
        }

        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:    int                             iTermId client browser
terminal id
 *                              int                             iSyncId
        client browser sync id
 *                              BOOL                    bInput   TRUE 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;
        char    szTmp[64];
        char    szW_Zip[26];
        char    szD_Zip[26];
        char    szC_Zip[26];
        char    szC_Phone[26];
        char    szTmpStr1[122];
        char    szTmpStr2[122];
        char    szTmpStr3[122];
        char    szTmpStr4[122];
        int             i;
        int             l;
        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=\"\">");
        else
        {
#ifdef PERF_MEASURE
                {
                        PECBINFO pEcbInfo =
SQLGetECB(Term.pClientData[iTermId].dbproc);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T6\" VALUE=\"%d\">", pEcbInfo->iTMms);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T7\" VALUE=\"%d\">", pEcbInfo->iSUTms);
                }
#endif
        }
        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:    int                             iTermId  client browser
terminal id
 *                              int                             iSyncId
        client browser sync id
 *                              BOOL                    bInput   TRUE 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;
        char    c_first[98];
        char    c_middle[14];
        char    c_last[98];
        int             i;

        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=\"\">");
        else
        {
#ifdef PERF_MEASURE
                {
                        PECBINFO pEcbInfo =
SQLGetECB(Term.pClientData[iTermId].dbproc);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T6\" VALUE=\"%d\">", pEcbInfo->iTMms);
                        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\" NAME=\"T7\" VALUE=\"%d\">", pEcbInfo->iSUTms);
                }
#endif
        }

        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_suppl
y_w_id,

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

        Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_quant
ity,

        Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_amoun
t,

        Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliv
ery_d.day,

        Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliv
ery_d.month,

        Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliv
ery_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:    int                             iTermId client browser
terminal id
 *                              int                             iSyncId
        client browser sync id
 *                              BOOL                    bInput   TRUE if form
is being constructed for input else FALSE
 *                              BOOL                    bSuccess 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  *pECB   passed in structure pointer
from inetsrv.
 *                              int
        iTermId client browser terminal id
 *                              int
        iSyncId client browser sync id
 *
 * RETURNS:             None
 *
 * COMMENTS:    None
 *
 */

static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK    *pECB, int iTermId,
int iSyncId)
{
        int                     iRc;
        int                     iError;
        PECBINFO pEcbInfo;
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  *pECB    passed in structure pointer
from inetsrv.
 *                               int
        iTermId client browser terminal id
 *                               int
        iSyncId client browser sync id
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId)
{
        int                     iRc;
        int                     iError;
        PECBINFO pEcbInfo;


        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  *pECB    passed in structure pointer
from inetsrv.
 *                               int
        iTermId client browser terminal id
```

```
 *                                int
        iSyncId client browser sync id
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId)
{
        int                     iRc;
        int                     iError;
        PECBINFO pEcbInfo;


        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  *pECB   passed in structure pointer
from inetsrv.
 *                                int
        iTermId client browser terminal id
 *                                int
        iSyncId clinet browser sync id
 *
 * RETURNS:              None
 *
 * COMMENTS:     None
 *
 */

static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId)
{
        char    szTmp[26];
        BOOL    bSuccess;

        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  *pECB   passed in structure pointer
from inetsrv.
 *                                  int
        iTermId client browser terminal id
 *                                  int
        iSyncId client browser sync id
 *
 * RETURNS:             None
 *
 * COMMENTS:    None
 *
 */

static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId)
{
        char            szTmp[26];
        int                     iRc;
        PECBINFO pEcbInfo;


        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:   LPSTR                   lpszQueryString          client
browser http command string
 *                                NEW_ORDER_DATA   *pNewOrderData
        pointer to new order data structure
 *
 * RETURNS:             int
                        error code indicating reason for failure
 *                              ERR_SUCCESS
                        new order input data successfully parsed
 *
 *
 * COMMENTS:    None
 *
 */

static int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA
*pNewOrderData)
{
        char    szTmp[26];
        char    szKey[26];
        int             i;
        short   items;
        BOOL    bCheck;

        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:    LPSTR                   lpszQueryString          client
browser http command string
```

```
 *                                PAYMENT_DATA     *pPaymentData
        pointer to payment data structure
 *
 * RETURNS:                       int
                          error code indicating reason for failure
 *                                     ERR_SUCCESS
                          all input data successfully parsed
 *
 * COMMENTS:     None
 *
 */

static int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA *pPaymentData)
{
        char    szTmp[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:   LPSTR                           lpszQueryString
        client browser http command string
 *                              ORDER_STATUS_DATA       *pOrderStatusData
        pointer to order status data structure
 *
 * RETURNS:             int
                        error code indicating reason for failure
 *                              ERR_SUCCESS
                        successfully parsed all required input data
 *
 * COMMENTS:    None
 *
 */
static int GetOrderStatusData(LPSTR lpszQueryString, ORDER_STATUS_DATA
*pOrderStatusData)
{
        char    szTmp[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)
{
        HKEY    hKey;
```

```
        DWORD   size;
        DWORD   type;
        char    szTmp[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:   short                   w_id                    warehouse id
 *                              short                   o_carrier_id
        carrier id
 *
 * RETURNS:             BOOL    FALSE   delivery information posted
successfully
 *                                              TRUE    error 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_TRANSACTION    deliveryTransaction;
        int                                             d;
        int                                             i;

        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                    *ptr    pointer to string to check.
 *
 * RETURNS:             BOOL    FALSE   if string is not all numeric
 *                                              TRUE    if string contains
only numeric characters i.e. '0' - '9'
 *
 * COMMENTS:    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    *szBuff Returned string information
 *                              char    *szStr  input string to be formatted.
 *                              int             iLen    Length 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;
}
```

File: 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
```

File: 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=perf - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to perf - Win32 Debug.
!ENDIF

!IF "$(CFG)" != "tpcc - Win32 Release" && "$(CFG)" != "tpcc - Win32 Debug" &&\
 "$(CFG)" != "tux_client - Win32 Release" && "$(CFG)" !=\
 "tux_client - Win32 Debug" && "$(CFG)" != "tux_server - Win32 Release" &&\
 "$(CFG)" != "tux_server - Win32 Debug" && "$(CFG)" != "dll - Win32 Release"
&&\
 "$(CFG)" != "dll - Win32 Debug" && "$(CFG)" != "perf - Win32 Release" &&\
 "$(CFG)" != "perf - 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="perf - 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 "tux_client - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "tux_client - Win32 Debug" (based on "Win32 (x86) External Target")
!MESSAGE "tux_server - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "tux_server - Win32 Debug" (based on "Win32 (x86) External Target")
!MESSAGE "dll - Win32 Release" (based on "Win32 (x86) External Target")
!MESSAGE "dll - Win32 Debug" (based on "Win32 (x86) External Target")
!MESSAGE "perf - Win32 Release" (based on "Win32 (x86) Console Application")
!MESSAGE "perf - 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 "tpcc - Win32 Release"

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

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

```
INTDIR=.\Release

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

CLEAN :
        -@erase

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

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

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

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


CLEAN :
        -@erase

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

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

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

ALL :

CLEAN :
        -@erase

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

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

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

```
INTDIR=.\tux_client\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 CFG=Release /f tux_server.mak"
# PROP Rebuild_Opt "/a"
# PROP Target_File "Release\tux_server\tpcc.exe"
# PROP Bsc_Name ""
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 CFG=Debug /f tux_server.mak"
# PROP Rebuild_Opt "/a"
# PROP Target_File "Debug\tux_server\tpcc.exe"
# PROP Bsc_Name ""
OUTDIR=.\tux_server\Debug
INTDIR=.\tux_server\Debug

ALL :

CLEAN :
        -@erase

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

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

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

ALL :
```

```
CLEAN :
        -@erase

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

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

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

ALL :

CLEAN :
        -@erase

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

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

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

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

CLEAN :
        -@erase "$(INTDIR)\perf.obj"
        -@erase "$(INTDIR)\tpcc_shm.obj"
        -@erase "$(INTDIR)\util.obj"
        -@erase "$(OUTDIR)\perf.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 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /YX /c
CPP_PROJ=/nologo /ML /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"\
 /Fp"$(INTDIR)/perf.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\perf\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)/perf.bsc"
BSC32_SBRS= \

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

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

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

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

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

CLEAN :
        -@erase "$(INTDIR)\perf.obj"
        -@erase "$(INTDIR)\tpcc_shm.obj"
        -@erase "$(INTDIR)\util.obj"
        -@erase "$(INTDIR)\vc40.idb"
        -@erase "$(INTDIR)\vc40.pdb"
        -@erase "$(OUTDIR)\perf.exe"
        -@erase "$(OUTDIR)\perf.ilk"
        -@erase "$(OUTDIR)\perf.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 /D "WIN32" /D "_DEBUG" /D "_CONSOLE" /YX
/c
CPP_PROJ=/nologo /MLd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_CONSOLE"\
 /Fp"$(INTDIR)/perf.pch" /YX /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
CPP_OBJS=.\perf\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)/perf.bsc"
BSC32_SBRS= \

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

"$(OUTDIR)\perf.exe" : "$(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.exe" :
   CD E:\home\bretm\src\win32\tpcc
   NMAKE /f tpcc.mak

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

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

!ENDIF

################################################################################
#
# Begin Project Dependency

# Project_Dep_Name "tux_client"

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

"tux_client - Win32 Release" :
   $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="tux_client - Win32 Release"

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

"tux_client - Win32 Debug" :
   $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="tux_client - Win32 Debug"

!ENDIF

# End Project Dependency
################################################################################
#
# Begin Project Dependency

# Project_Dep_Name "tux_server"

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

"tux_server - Win32 Release" :
   $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="tux_server - Win32 Release"

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

"tux_server - Win32 Debug" :
   $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="tux_server - Win32 Debug"

!ENDIF

# End Project Dependency
################################################################################
#
# Begin Project Dependency
```

```
# Project_Dep_Name "dll"

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

"dll - Win32 Release" :
   $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="dll - Win32 Release"

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

"dll - Win32 Debug" :
   $(MAKE) /$(MAKEFLAGS) /F ".\tpcc.mak" CFG="dll - Win32 Debug"

!ENDIF

# End Project Dependency
# End Target
################################################################################
#
# Begin Target

# Name "tux_client - Win32 Release"
# Name "tux_client - Win32 Debug"

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

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

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

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

!ENDIF

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

# Name "tux_server - Win32 Release"
# Name "tux_server - Win32 Debug"

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

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

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

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

!ENDIF

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

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

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

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

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

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

!ENDIF

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

SOURCE=.\src\tux_trans.c

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

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

!ENDIF

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

# Name "perf - Win32 Release"
# Name "perf - Win32 Debug"

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

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

!ENDIF

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

SOURCE=.\src\perf.c
DEP_CPP_PERF_=\
        ".\src\tpcc_shm.h"\
        ".\src\util.h"\


"$(INTDIR)\perf.obj" : $(SOURCE) $(DEP_CPP_PERF_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


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

SOURCE=.\src\tpcc_shm.c
DEP_CPP_TPCC_=\
        ".\src\tpcc_shm.h"\
        ".\src\util.h"\


"$(INTDIR)\tpcc_shm.obj" : $(SOURCE) $(DEP_CPP_TPCC_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


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

File: tpcc1.def

```
LIBRARY TPCC.DLL

EXPORTS

        GetExtensionVersion      @1
        HttpExtensionProc        @2
```

File: 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 c 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
```

File: tpcc2.def

```
LIBRARY TPCC2.DLL

EXPORTS

        GetExtensionVersion      @1
        HttpExtensionProc        @2
```

File: 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 c 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
```

File: tpcc_real.h

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


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


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

```
#define ORDER_STATUS_FORM                               6
        //order status id
#define STOCK_LEVEL_FORM                                7
        //stock level form id

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


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

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

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

//This structure defines the data necessary to keep distinct for each terminal
or client connection.
typedef struct _CLIENTDATA
{
        int                     inUse;                                  //in
use flag allows client entries to be reused
        int                     w_id;
        //warehouse id assigned at welcome form
        int                     d_id;
        //district id assigned at welcome form
        PDBPROCESS      dbproc;                                 //dblib
connection pointer
        int                     spid;
        //spid assigned from dblib
        int                     iSyncId;
        //syncronization id
        int                     iTickCount;
        //time of last access;
        int                     iTermId;                        //terminal
id of http stream connection

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

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

typedef CLIENTDATA *PCLIENTDATA;                  //pointer to client structure

//This structure is used to define the operational interface for terminal id
support
typedef struct _TERM
{
        int                     iAvailable;
                //total allocated terminal array entries
        int                     iNext;
                //next available terminal array element
        int                     iMasterSyncId;
        //syncronization id
        BOOL            bInit;
        //structure has been initialized flag
        CLIENTDATA      *pClientData;
        //pointer to allocated client data
        void            (*Init)(void);
        //API to initialize this structure
        int                     (*Allocate)(void);
        //API to allocate a new terminal entry array id returned
        void            (*Restore)(void);
        //API to free terminal data
```

```
        int                         (*Add)(EXTENSION_CONTROL_BLOCK *pECB, char
*pQueryString);  //API to add a terminal id to array, this context will

                            //be passed from the browser to the tpcc.dll in the

                            //TERMID= key in the HTTP string.
        void                (*Delete)(EXTENSION_CONTROL_BLOCK *pECB, int id);
        //API to free resources used by a terminal array entry
} TERM;

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



//function prototypes

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

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

extern char szErrorLogPath[256];
```

```
extern EXTENSION_CONTROL_BLOCK    *gpECB;
```

File: tpcc_shm.c

```
#include <windows.h>
#include <memory.h>

#include "tpcc_shm.h"
#include "util.h"

static HANDLE hMapping;
static  tpcc_perf_shm *pShm=NULL;

volatile tpcc_perf_shm *
MapTPCCPerfShm(void)
{
        PSECURITY_DESCRIPTOR pSD;
        SECURITY_ATTRIBUTES  sa;

        if (pShm)
        {
                Log("attempt to reopen tux shm\n");
                return NULL;
        }


        /*
         * all the security stuff is because inetinfo runs as a service.
         * See the Knowlege base article #Q106387 for the gory details
         */

        pSD = (PSECURITY_DESCRIPTOR) LocalAlloc(LPTR,
SECURITY_DESCRIPTOR_MIN_LENGTH);
        if (pSD == NULL)
        {
                Log("LocalAlloc failed with %d\n", GetLastError());
                return NULL;
        }

        if (!InitializeSecurityDescriptor(pSD, SECURITY_DESCRIPTOR_REVISION))
        {
                Log("InitializeSecurityDescriptor failed with %d\n",
                        GetLastError());
                return NULL;
        }
        // Add a NULL DACL to the security descriptor..
        if (!SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL, FALSE))
        {
                Log("SetSecurityDescriptor Failed with %d\n", GetLastError());
                return NULL;
        }

        sa.nLength = sizeof(sa);
        sa.lpSecurityDescriptor = pSD;
        sa.bInheritHandle = TRUE;

        hMapping = CreateFileMapping((HANDLE)0xffffffff, &sa, PAGE_READWRITE,
                        0, sizeof(struct tpcc_perf_shm), "TPCC Shared Perf
Data");

        if (hMapping == NULL)
                Log("CreateFileMapping failed with error %d\n",
                        GetLastError());
        else
        {
                DWORD dwError = GetLastError();
                pShm = (tpcc_perf_shm *)MapViewOfFile(hMapping,
FILE_MAP_ALL_ACCESS, 0, 0, 0);
                if (!pShm)
                        CloseHandle(hMapping);
                else if (dwError != ERROR_ALREADY_EXISTS && pShm)
                {
                        memset(pShm, 0, sizeof(*pShm));
                }
        }

        Log("MapTPCCPerfShm Return 0x%x\n", pShm);

        return pShm;
}

void
UnMapTPCCPerfShm(volatile tpcc_perf_shm *pFree)
{
        if (pShm && pFree != pShm)
```

```
        {
                Log("pFree (0x%x) != pShm (0x%x)\n");
                return;
        }

        if (!pShm)
                return;

        UnmapViewOfFile(pShm);
        pShm = NULL;
        CloseHandle(hMapping);
}
```

File: tpcc_shm.h

```
#ifndef TPCC_SHM_H_INCLUDED
#define TPCC_SHM_H_INCLUDED

#define MAX_TUX_SERVERS 1024

/*
 * This file contains the definitions for the mapped file shared memory which
 * is used to store performance information for the tpcc benchmark components.
 *
 * In order to prevent the need for using expensive kernel synchronization
 * mechanisms, an alternate scheme is used.  For each data structure to be
 * protected, a counter is incremented before any updates occur and a different
 * counter is updated after the updates have occured.  Since all of the
structures have
 * only one writer, this works.
 * In order to prevent deadlock, the following convention applies:
 *     The generation counts are updated in substructures before being updated
 *     in a higher level structure.  For example, it is legal for a tuxedo
 *     server to increment tux_perf_info->lUpdatesBegun and then check for
 *     (and wait for) the data in tux_shm
 */

/*
 * the definitions for the tuxedo performance instrumentation
 */

typedef struct tux_server_perf_info
{
        long lUpdatesBegun;
        __int64 iIdleTicks;
        __int64 iBusyTicks;
        __int64 iTransactions;
        long lBusy;
        LARGE_INTEGER liLast;
        long lUpdatesCompleted;
} tux_server_perf_info;

typedef struct tux_perf_header
{
        long lUpdatesBegun;
        long lUpdatesCompleted;
        long iTuxCount;
} tux_perf_header;

typedef struct tux_perf_info
{
        tux_perf_header Header;
        tux_server_perf_info Servers[MAX_TUX_SERVERS];

} tux_perf_info;

/*
 * the definitions for the tpcc.dll instrumentation
 */

typedef struct t_DLL_PERF_COUNTER
{
        ULONG ulUpdatesBegun;
        CRITICAL_SECTION cs;
        __int64 iCount;
        __int64 iQ1Time;
        __int64 iSUTTime;
        __int64 iQ2Time;
        __int64 iTUXTime;
        ULONG ulUpdatesCompleted;
} DLL_PERF_COUNTER;


typedef enum eTRANS {NEW_ORDER_TRANS, ORDER_STATUS_TRANS, STOCK_LEVEL_TRANS,
PAYMENT_TRANS, DELIVERY_TRANS, TRANS_TYPE_COUNT} TRANS;
```

```
typedef struct t_DLL_PERF_INFO
{
        DLL_PERF_COUNTER Counters[TRANS_TYPE_COUNT];
} DLL_PERF_INFO;

/*
 * the structure in shared memory which combines all the performance
 * information
 */
typedef struct tpcc_perf_shm
{
        tux_perf_info TuxInfo;
        DLL_PERF_INFO DLLInfo;
} tpcc_perf_shm;

tpcc_perf_shm volatile * MapTPCCPerfShm(void);
void UnMapTPCCPerfShm(tpcc_perf_shm volatile * pFree);

#endif
```

File: 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
```

File: trans.h

```
/*      FILE:           TRANS.H
 *                              Microsoft TPC-C Kit Ver. 3.00.000
 *                              Audited 08/23/96 By 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

        #define DEFCLPACKSIZE                           4096
        #define DEADLOCKWAIT                            10

        // String length constants
```

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

// transaction structures

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

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

typedef struct
{
        short                           w_id;
        short                           d_id;
        long                            c_id;
        short                           c_d_id;
        short                           c_w_id;
        double                          h_amount;
#ifdef USE_ODBC
        TIMESTAMP_STRUCT  h_date;
#else
        DBDATEREC               h_date;
#endif
        char                            w_street_1[ADDRESS_LEN+1];
        char                            w_street_2[ADDRESS_LEN+1];
        char                            w_city[ADDRESS_LEN+1];
        char                            w_state[STATE_LEN+1];
        char                            w_zip[ZIP_LEN+1];
        char                            d_street_1[ADDRESS_LEN+1];
        char                            d_street_2[ADDRESS_LEN+1];
```

```
        char                            d_city[ADDRESS_LEN+1];
        char                            d_state[STATE_LEN+1];
        char                        d_zip[ZIP_LEN+1];
        char                            c_first[FIRST_NAME_LEN+1];
        char                            c_middle[MIDDLE_NAME_LEN +
1];
        char                    c_last[LAST_NAME_LEN+1];
        char                            c_street_1[ADDRESS_LEN+1];
        char                            c_street_2[ADDRESS_LEN+1];
        char                            c_city[ADDRESS_LEN+1];
        char                            c_state[STATE_LEN+1];
        char                            c_zip[ZIP_LEN+1];
        char                            c_phone[PHONE_LEN+1];
#ifdef USE_ODBC
        TIMESTAMP_STRUCT  c_since;
#else
        DBDATEREC                   c_since;
#endif
        char                            c_credit[CREDIT_LEN+1];
        double                          c_credit_lim;
        double                          c_discount;
        double                          c_balance;
        char                            c_data[200+1];
        long                            num_deadlocks;
        char
execution_status[STATUS_LEN];
} PAYMENT_DATA;

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

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

File: tux.h

```
#ifndef TUX_H_INCLUDED
#define TUX_H_INCLUDED

#define SERVICE_CHARS 32

#define PERF_MEASURE

#if defined(_DEBUG) && !defined(PERF_MEASURE)
#define PERF_MEASURE
#endif

typedef union
{
        NEW_ORDER_DATA          NewOrderData;
        PAYMENT_DATA            PaymentData;
        ORDER_STATUS_DATA       OrderStatusData;
        DELIVERY_DATA           DeliveryData;
        STOCK_LEVEL_DATA StockLevelData;
        char                    ErrorMsg[4000]; // ack!!
} TRANS_DATA;

typedef struct
{
        int TermId;
        int SyncId;
        int bDeadlock;
        int bFailed;
        short DeadlockRetry;
        int Error;
        int Return;
#ifdef PERF_MEASURE
        LARGE_INTEGER liStartCounter;
        LARGE_INTEGER liEndCounter;
#endif
#ifdef TRANS_SEQ
        int Seq;
#endif
        // 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 DATA_HEADER_SIZE(p)       ((DWORD)(((char *)&(p)->Trans) - ((char
*)(p))))
#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
```

File: tux_client.c

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <direct.h>
#ifdef _DEBUG
#include <time.h>
#endif
```

```
#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 "util.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_FILE="c:\\temp\\tpcc_logs\\tux_client\\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
        Log("about to tpcall Service %s, bufsize=%d\n", Service, BufSize);
#endif
        if (tpcall(Service, TuxBuffer, BufSize, &TuxBuffer, pnRead, TPNOTIME)
== -1)
        {
                fprintf(stderr, "TuxTransaction: tpcall failed, tperrno=%d\n",
tperrno);
                return FALSE;
        }

#ifdef _DEBUG
        Log("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, sizeof(msg.Data),
&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];

        if (!mkpath(LOG_FILE))
        {
                fprintf(stderr, "mkpath failed for %s\n", LOG_FILE);
                exit(1);
        }

        sprintf(buf, LOG_FILE, 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
        Log("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
                Log("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());
        else
        {

                if (ClientNumber >= InitialCreate-1)
                        StartAnother(argv[0], ClientNumber + 1,
InitialCreate);

                HandleTransactions(hPipe);

                CloseHandle(hPipe);
        }

        TuxCleanup();

        return 0;
}
```

File: 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[]="c:\\temp\\tpcc_logs\\tux_server.txt";
DBPROCESS *pdbproc;
char *Server=NULL;;
char *Database="tpcc";
char *User="sa";
char *Password="";
int spId;
TUX_DATA data;
TERM Term;
EXTENSION_CONTROL_BLOCK *gpECB=NULL;

#ifdef PERF_MEASURE
LARGE_INTEGER CounterFrequencyMS;
#include "tpcc_shm.h"
tpcc_perf_shm *pShm=NULL;
tux_server_perf_info *pPerfInfo=NULL;
#endif

/* #define PRINT_TRANS /* */

void TuxLog(char *format, ...)
```

```
{
        va_list args;
        char buf[4096];
        int len;

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

        userlog(buf);
}

static int
ExceptionFilter(int iException, int iLine, char *pFile)
{
        TuxLog("thread %d caught exception 0x%x from line %d in file %s\n",
                GetCurrentThreadId(), iException, iLine, pFile);

        return EXCEPTION_EXECUTE_HANDLER;
}

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

BOOL IsValidTermId(int TermId)
{
        return FALSE;
}

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

        __try
        {
                mkpath(szErrorLogPath);

                TuxLog("starting the tuxedo TPCC server");

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

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

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

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

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

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

                pShm = (tpcc_perf_shm *)MapTPCCPerfShm();
                pPerfInfo = pShm->TuxInfo.Servers +
InterlockedIncrement(&pShm->TuxInfo.Header.iTuxCount);
                TuxLog("pPerfInfo=0x%x, my num=%d\n", pPerfInfo, pPerfInfo -
pShm->TuxInfo.Servers);
                QueryPerformanceCounter(&pPerfInfo->liLast);
#endif
        }
```

```
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                TuxLog("caught an exception in tpsvrinit\n");
        }
        return 0;
}

void
tpsvrdone(void)
{
        __try
        {
                TuxLog("shuttting down the tuxedo TPCC server");
                free(Server);
                SQLCloseConnection(NULL, pdbproc);
                SQLCleanup();
#ifdef PERF_MEASURE
                UnMapTPCCPerfShm(pShm);
#endif
        }
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                TuxLog("caught an exception in tpsvrdone\n");
        }
}

void
start_trans(void)
{
#ifdef PERF_MEASURE
        InterlockedIncrement(&pPerfInfo->lUpdatesBegun);

        QueryPerformanceCounter(&data.liStartCounter);

        /* account for our idle time and mark us busy */
        pPerfInfo->iIdleTicks += data.liStartCounter.QuadPart -
                                 pPerfInfo->liLast.QuadPart;
        pPerfInfo->lBusy = 1;
        pPerfInfo->liLast = data.liStartCounter;
        InterlockedIncrement(&pPerfInfo->lUpdatesCompleted);
#endif

#ifdef TRANS_SEQ
        TuxLog("B 0x%x\n", data.Seq);
#endif
}

void
end_trans(void)
{
#ifdef PERF_MEASURE
        InterlockedIncrement(&pPerfInfo->lUpdatesBegun);

        QueryPerformanceCounter(&data.liEndCounter);

        /* account for the busy time and mark us idle */

        pPerfInfo->iBusyTicks += data.liEndCounter.QuadPart -
                                 pPerfInfo->liLast.QuadPart;
        pPerfInfo->lBusy = 0;
        pPerfInfo->iTransactions++;
        pPerfInfo->liLast = data.liEndCounter;
        InterlockedIncrement(&pPerfInfo->lUpdatesCompleted);
#endif

#ifdef TRANS_SEQ
        TuxLog("E 0x%x\n", data.Seq);
#endif
}

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

        __try
        {

                memcpy(&data, rqst->data, size);

                start_trans();

#ifdef PRINT_TRANS
                if (data.Trans.NewOrderData.c_id)
                        TuxLog("Start NEW_ORDER for customer %d, size=%d
data=0x%x\n",data.Trans.NewOrderData.c_id, size, rqst->data);
                else
                        TuxLog("Start NEW_ORDER for customer %s size=%d
data=0x%x\n",data.Trans.NewOrderData.c_last, size, rqst->data);
```

```
#endif

                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);

                        /*
                         * we reallocate the buffer here.  tpreturn will
notice the buffer has changed
                         * and will automatically free the buffer we were sent
                         */
                        if (size > rqst->len)
                                rqst->data = (char *) tpalloc("CARRAY", NULL,
size);

                        strcpy(data.Trans.ErrorMsg,
Term.pClientData[0].szBuffer);
                }

#ifdef PRINT_TRANS
                TuxLog("End NEWORDER, in %I64dms bFailed=%d\n",
                        (data.liEndCounter.QuadPart -
data.liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart,
                        data.bFailed);
#endif

                end_trans();
                memcpy(rqst->data, &data, size);
                tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
        }
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                TuxLog("caught an exception in NEW_ORDER\n");
        }
}

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

        __try
        {
                memcpy(&data, rqst->data, size);

                start_trans();
#ifdef PRINT_TRANS
                TuxLog("Start STOCK_LEVEL data=0x%x\n", rqst->data);
#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);

                        /*
                         * we reallocate the buffer here.  tpreturn will
notice the buffer has changed
                         * and will automatically free the buffer we were sent
                         */
                        if (size > rqst->len)
                                rqst->data = (char *) tpalloc("CARRAY", NULL,
size);

                        strcpy(data.Trans.ErrorMsg,
Term.pClientData[0].szBuffer);
                }

#ifdef PRINT_TRANS
                TuxLog("End STOCK_LEVEL, in %I64dms bFailed=%d\n",
                        (data.liEndCounter.QuadPart -
data.liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart,
                        data.bFailed);
#endif

                end_trans();
                memcpy(rqst->data, &data, size);
                tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
        }
```

```
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                TuxLog("caught an exception in STOCK_LEVEL\n");
        }
}

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

        __try
        {
                memcpy(&data, rqst->data, size);
                start_trans();

#ifdef PRINT_TRANS
                if (data.Trans.PaymentData.c_id)
                        TuxLog("Start PAYMENT for customer %d
data=0x%x\n",data.Trans.PaymentData.c_id, rqst->data);
                else
                        TuxLog("Start PAYMENT for customer %s
data=0x%x\n",data.Trans.PaymentData.c_last, rqst->data);
#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);

                        /*
                         * we reallocate the buffer here.  tpreturn will
notice the buffer has changed
                         * and will automatically free the buffer we were sent
                         */
                        if (size > rqst->len)
                                rqst->data = (char *) tpalloc("CARRAY", NULL,
size);
                        strcpy(data.Trans.ErrorMsg,
Term.pClientData[0].szBuffer);
                }

#ifdef PRINT_TRANS
                TuxLog("End PAYMENT, in %I64dms bFailed=%d\n",
                        (data.liEndCounter.QuadPart -
data.liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart,
                        data.bFailed);
#endif
                end_trans();
                memcpy(rqst->data, &data, size);
                tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
        }
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                TuxLog("caught an exception in PAYMENT\n");
        }

}

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

        __try
        {
                memcpy(&data, rqst->data, size);
                start_trans();
#ifdef PRINT_TRANS
                TuxLog("Start ORDER_STATUS data=0x%x\n", rqst->data);
#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);
```

```
                        /*
                         * we reallocate the buffer here.  tpreturn will
notice the buffer has changed
                         * and will automatically free the buffer we were sent
                         */
                        if (size > rqst->len)
                                rqst->data = (char *) tpalloc("CARRAY", NULL,
size);

                        strcpy(data.Trans.ErrorMsg,
Term.pClientData[0].szBuffer);
                }

#ifdef PRINT_TRANS
                TuxLog("End ORDER_STATUS, in %I64dms bFailed=%d\n",
                        (data.liEndCounter.QuadPart -
data.liStartCounter.QuadPart)/CounterFrequencyMS.QuadPart,
                        data.bFailed);
#endif

                end_trans();
                memcpy(rqst->data, &data, size);
                tpreturn(TPSUCCESS, 0, rqst->data, size, 0);
        }
        __except(ExceptionFilter(GetExceptionCode(), __LINE__, __FILE__))
        {
                TuxLog("caught an exception in ORDER_STATUS\n");
        }
}
```

File: 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

OUT_PATH= .
SRCDIR          = ..\Src
OBJDIR          = $(OUT_PATH)\$(CFG)
OUTDIR          = $(OUT_PATH)\Bin
TUX              = d:\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" "$(OBJDIR)\tpcc_shm.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)\tpcc_shm.obj":
        cl.exe $(FLAGS) /I$(TUXINCLUDE) /Fd$(OBJDIR)\tux_server.pdb
/Fo$(OBJDIR)\tpcc_shm.obj /c "$(SRCDIR)\tpcc_shm.c"

"$(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)"
```

File: tux_sql.c

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <time.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"
#include "tux_trans.h"


BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
DBPROCESS **dbproc, char *server, char *database, char *user, char *password,
char *app, int *spid)
{
        PECBINFO pEcbInfo;

        //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 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;
}
```

File: tux_trans.c

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

#include <atmi.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"
#include "tux_trans.h"

static  CRITICAL_SECTION         CriticalSection;

void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr);

volatile int ThreadCount;
volatile DWORD TlsIndex=0xffffffff;

TPINIT * volatile pTPInit=NULL;

#ifdef PERF_MEASURE

#include "tpcc_shm.h"
volatile tpcc_perf_shm *pPerfShm;

LARGE_INTEGER liCounterFrequencyMS;

DLL_PERF_COUNTER *Counters=NULL;

#endif

/*
 * 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
 */

BOOL DoInit(int ClientNumber)
{

#ifdef _DEBUG
```

```
        fprintf(stderr, "DoInit begins for thread %d, pTPInit=0x%x\n",
ThreadCount, pTPInit);
#endif

        if (!pTPInit)
        {
                pTPInit = (TPINIT *)tpalloc("TPINIT", NULL, sizeof(TPINIT));

                if (!pTPInit)
                {
                        fprintf(stderr, "tpalloc of pTPInit failed for thread
%d\n",
                                ThreadCount);
                        return FALSE;
                }

                pTPInit->flags |= TPMULTICONTEXTS;
        }

        _snprintf(pTPInit->cltname, sizeof(*pTPInit->cltname), "cl%d",
ClientNumber);

        if (tpinit(pTPInit) == -1)
        {
                fprintf(stderr, "tpinit failed for thread %d, tperrno=%d\n",
                        ThreadCount, tperrno);
                return FALSE;
        }

#ifdef _DEBUG
        fprintf(stderr, "tpinit succeeded for thread %d\n", ThreadCount);
#endif
        TlsSetValue(TlsIndex, (void *)TRUE);

        return TRUE;
}

BOOL SQLThreadAttach(void)
{
        BOOL bReturn = TRUE;

#ifdef _DEBUG
        fprintf(stderr, "thread %d about to attach\n", GetCurrentThreadId());
#endif
        EnterCriticalSection(&CriticalSection);

        if (!DoInit(ThreadCount))
        {
                Log("thread %d unable to Initialize in SQLThreadAttach()\n",
GetCurrentThreadId());
                bReturn=FALSE;
        }

        if (bReturn)
                ThreadCount++;

        LeaveCriticalSection(&CriticalSection);
#ifdef _DEBUG
        fprintf(stderr, "thread %d attach returns %d\n", GetCurrentThreadId(),
bReturn);
#endif

        return bReturn;
}

BOOL SQLThreadDetach(void)
{

#ifdef _DEBUG
        fprintf(stderr, "thread %d detaching\n", GetCurrentThreadId());
#endif
        EnterCriticalSection(&CriticalSection);
        ThreadCount--;
        LeaveCriticalSection(&CriticalSection);

        tpterm();

        return TRUE;
}

BOOL SQLInit(void)
{
        // Perform one time initialization.
#ifdef PERF_MEASURE
        int i;
        LARGE_INTEGER liFrequency;

        pPerfShm = MapTPCCPerfShm();

        QueryPerformanceFrequency(&liFrequency);
```

```
        liCounterFrequencyMS.QuadPart = liFrequency.QuadPart/1000;

        Counters = pPerfShm->DLLInfo.Counters;

        for(i=0;i<TRANS_TYPE_COUNT;i++)
                InitializeCriticalSection(&Counters[i].cs);

#endif

#ifdef _DEBUG
        fprintf(stderr, "SQLInit() called\n");
#endif
        InitializeCriticalSection(&CriticalSection);

        TlsIndex = TlsAlloc();

        if (TlsIndex == 0xffffffff)
        {
                Log("TlsAlloc Failed\n");
                return FALSE;
        }

        return SQLThreadAttach();
}

void SQLCleanup(void)
{
#ifdef _DEBUG
        fprintf(stderr, "SQLCleanup() called\n");
#endif
#ifdef PERF_MEASURE
        {
                int i;
                for(i=0;i<TRANS_TYPE_COUNT;i++)
                        DeleteCriticalSection(&Counters[i].cs);
        }
        UnMapTPCCPerfShm(pPerfShm);
#endif
        SQLThreadDetach();
        tpfree((char *)pTPInit);
        pTPInit = NULL;
        DeleteCriticalSection(&CriticalSection);
}
```

```
static int
ExceptionFilter(int iException)
{
        Log("thread %d caught exception 0x%x\n", GetCurrentThreadId(),
iException);

        return EXCEPTION_EXECUTE_HANDLER;
}

void
LogTransData(char *Service, TUX_DATA *pData)
{
        if (!strcmp(Service, "PAYMENT"))
        {
                if (pData->Trans.PaymentData.c_id)
                        Log("Thread %d Payment for Customer=%d\n",
                                GetCurrentThreadId(),
                                pData->Trans.PaymentData.c_id);
                else
                        Log("Thread %d Payment for Customer=%s\n",
                                GetCurrentThreadId(),
                                pData->Trans.PaymentData.c_last);
        }
        else if (!strcmp(Service, "NEW_ORDER"))
        {
                if (pData->Trans.NewOrderData.c_id)
                        Log("Thread %d NewOrder for Customer=%d\n",
                                GetCurrentThreadId(),
                                pData->Trans.NewOrderData.c_id);
                else
                        Log("Thread %d NewOrder for Customer=%s\n",
                                GetCurrentThreadId(),
                                pData->Trans.NewOrderData.c_last);
        }
        else
        {
                Log("Thread %d %s\n", GetCurrentThreadId(), Service);
        }
}

int TuxTransaction(char *Service, EXTENSION_CONTROL_BLOCK  *pECB,
        int TermId, int SyncId, DBPROCESS *dbproc, short DeadlockRetry, void
*Data, long BufSize)
{
        TUX_DATA *pData;
```

```
        DWORD nRead;
        PECBINFO pECBInfo = (PECBINFO)dbproc; // forgive them them, for they
know not what they do...
        int iReturn;
#ifdef PERF_MEASURE
        LARGE_INTEGER liStartCounter, liEndCounter;
        DLL_PERF_COUNTER *pCounter;
        __int64 iSutms, iTotalms, iQ1ms, iQ2ms;
#endif

        // we are pessimistic here
        pECBInfo->bFailed = TRUE;

        if (!TlsGetValue(TlsIndex))
        {
                if (!SQLThreadAttach())
                {
                        Log("thread %d attach failed\n",
GetCurrentThreadId());
                        return -1;
                }
        }

        pData = (TUX_DATA *) tpalloc("CARRAY", NULL, sizeof(TUX_DATA));

        if (!pData)
        {
                Log("thread %d tpalloc() failed\n", GetCurrentThreadId());
                return -1;
        }

#ifdef _DEBUG
        Log("thread %d TuxTransaction pData=0x%x\n", GetCurrentThreadId(),
pData);
        Log("thread %d BufSize=%d, sizeof(TUX_DATA)=%d
sizeof(TRANS_DATA)=%d\n", GetCurrentThreadId(), BufSize, sizeof(TUX_DATA),
sizeof(TRANS_DATA));
#endif

        // fill the struct to ship to tux
        memcpy(&pData->Trans, Data, BufSize);
        pData->TermId = TermId;
        pData->SyncId = SyncId;
        pData->DeadlockRetry = DeadlockRetry;
        pData->Error = FALSE;

#ifdef TRANS_SEQ
        {
                char *p = strstr(pECB->lpszQueryString, "SEQ=");

                if (p)
                        pData->Seq = strtoul(p+4, NULL, 0);
        }
#endif
        BufSize += DATA_HEADER_SIZE(pData); // Send the headers too

#ifdef _DEBUG
        Log("thread %d about to tpcall Service %s, bufsize=%d\n",
GetCurrentThreadId(), Service, BufSize);

        LogTransData(Service, pData);
#endif

#ifdef PERF_MEASURE
        QueryPerformanceCounter(&liStartCounter);
#endif

        __try
        {
#ifdef TRANS_SEQ
                Log("C 0x%x\n", pData->Seq);
#endif
                if (tpcall(Service, (char *)pData, BufSize, &(char *)pData,
&nRead, 0) == -1)
                {
                        Log("thread %d tpcall failed tperrno=%d\n",
GetCurrentThreadId(), tperrno);
                        tpfree((char *)pData);

                        return -1;
                }
#ifdef TRANS_SEQ
                Log("R 0x%x\n", pData->Seq);
#endif
        }
        __except(ExceptionFilter(GetExceptionCode()))
        {
                Log("thread %d handling an exception from tpcall.  Service=%s,
pData=0x%x, BufSize=%d\n", GetCurrentThreadId(), Service, pData, BufSize);
                // we nest the try here, since it is fairly likely that
                // we got here because pData was bad in the tpcall.
```

```
                __try
                {
                        LogTransData(Service, pData);
                }
                __except(ExceptionFilter(GetExceptionCode()))
                {
                        Log("thread %d LogTransData() caused an exception\n",
GetCurrentThreadId());
                }
                tpfree((char *)pData);
                return -1;
        }

#ifdef PERF_MEASURE
        QueryPerformanceCounter(&liEndCounter);
        iTotalms = (liEndCounter.QuadPart -
liStartCounter.QuadPart)/liCounterFrequencyMS.QuadPart;
        iSutms = (pData->liEndCounter.QuadPart - pData-
>liStartCounter.QuadPart)/liCounterFrequencyMS.QuadPart,
        iQ1ms = (pData->liStartCounter.QuadPart -
liStartCounter.QuadPart)/liCounterFrequencyMS.QuadPart;
        iQ2ms = (liEndCounter.QuadPart - pData-
>liEndCounter.QuadPart)/liCounterFrequencyMS.QuadPart;

        switch (*Service)
        {
                case 'N':
                        pCounter = Counters + NEW_ORDER_TRANS;
                        break;
                case 'P':
                        pCounter = Counters +  PAYMENT_TRANS;
                        break;
                case 'O':
                        pCounter = Counters +  ORDER_STATUS_TRANS;
                        break;
                case 'S':
                        pCounter = Counters +  STOCK_LEVEL_TRANS;
                        break;
                default:
                        Log("Unable to determine Trans for %s\n", Service);
                        break;
        }

        InterlockedIncrement(&pCounter->ulUpdatesBegun);
        EnterCriticalSection(&pCounter->cs);

        pCounter->iCount++;
        pCounter->iSUTTime += iSutms;
        pCounter->iTUXTime += iTotalms - iSutms;
        pCounter->iQ1Time += iQ1ms;
        pCounter->iQ2Time += iQ2ms;

        LeaveCriticalSection(&pCounter->cs);
        InterlockedIncrement(&pCounter->ulUpdatesCompleted);

        pECBInfo->iSUTms = (int)iSutms;
        pECBInfo->iTMms = (int)(iTotalms - iSutms);

#ifdef _DEBUG
        Log("thread %d end %s in %I64dms (tux=%I64dms, sut=%I64dms), %d bytes
pData=0x%x, bFailed=%d\n", GetCurrentThreadId(), Service,
                                iTotalms, iTotalms - iSutms, iSutms, nRead,
pData, pData->bFailed);
#endif
#endif

        if (nRead < (DWORD)BufSize)
        {
                Log("thread %d nRead(%d) < BufSize(%d)\n",
GetCurrentThreadId(), nRead, BufSize);
                tpfree((char *)pData);
                return -1;
        }
        else if (nRead > (DWORD)BufSize)
        {
#ifdef _DEBUG
                Log("thread %d nRead(%d) > BufSize(%d)\n",
GetCurrentThreadId(), nRead, BufSize);
#endif
        }

        BufSize -= DATA_HEADER_SIZE(pData); // Ignore the headers now

        if (pData->Error)
        {
#ifdef _DEBUG
                Log("thread %d pData->Error set, ErrorMsg=%s\n",
GetCurrentThreadId(), pData->Trans.ErrorMsg);
#endif
                WriteZString(pECB, pData->Trans.ErrorMsg);
```

```
        }

        // patch things up so the upper levels don't know this went
        // through tux

        pECBInfo->iTermId = TermId;
        pECBInfo->iSyncId = SyncId;
        pECBInfo->bDeadlock = pData->bDeadlock;
        pECBInfo->bFailed = pData->bFailed;


        if (pData->Error)
                Log("thread %d, Term=%d pData->Error set, ErrorMsg=%s\n",
GetCurrentThreadId(), TermId, pData->Trans.ErrorMsg);
        if (pData->bFailed)
                Log("thread %d Term=%d pData->bFailed is True\n",
GetCurrentThreadId(), TermId);

#ifdef _DEBUG
        // it is Ok to get Return == 0, because some transactions are supposed
        // to fail.  Only note if for _DEBUG
        if (!pData->Return)
                Log("Thread %d pData->Return == FALSE\n",
GetCurrentThreadId());
#endif

        memcpy(Data, &pData->Trans, BufSize);

        iReturn = pData->Return;

        tpfree((char *)pData);

        return iReturn;
}
```

File: tux_trans.h

```
#ifndef __TUX_INIT_H_INCLUDED
#define __TUX_INIT_H_INCLUDED

int TuxTransaction(char *Service, EXTENSION_CONTROL_BLOCK  *pECB,
        int TermId, int SyncId, DBPROCESS *dbproc, short DeadlockRetry,
        void *Data, long BufSize);

#endif
```

File: tux_trans_client.c

```
#include <windows.h>
#include <stdio.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"
#include "tux_trans.h"

const int ARGSIZE=1024;
const int PIPE_BUF_SIZE=4096;

static  CRITICAL_SECTION         CriticalSection;

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)
        {
                fprintf(stderr, "SQLThreadattach failed for thread %d\n",
pData->ThreadNumber);
                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 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))
        {
                char error_buf[1024];
                ErrorString(error_buf, sizeof(error_buf), GetLastError());
                fprintf(stderr, "Tuxtransaction: WritePipe Failed [%s]\n",
error_buf);
                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))
        {
                char error_buf[1024];
                ErrorString(error_buf, sizeof(error_buf), GetLastError());
                fprintf(stderr, "Tuxtransaction: ReadPipe Failed [%s]\n",
error_buf);
                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
        if (msg.Data.Error)
                fprintf(stderr, "Term=%d msg.Error set, ErrorMsg=%s\n",
TermId, msg.Data.Trans.ErrorMsg);
        if (msg.Data.bFailed)
                fprintf(stderr, "Term=%d msg.Data.bFailed is True\n", TermId);

#endif
        memcpy(Data, &msg.Data.Trans, BufSize);
```

```
        return msg.Data.Return;
}
```

File: tuxedo_threads.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 $(SQL_LOC)
!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

OUT_PATH= .
SRCDIR          = ..\..\Src
OBJDIR          = $(OUT_PATH)\$(CFG)
OUTDIR          = $(OUT_PATH)\Bin
ODBC             = \progra~1\msdev

DBLIB           = $(SQL_LOC)
DBLIBINC        = $(DBLIB)\INCLUDE
ODBCINCDIR       = $(ODBC)\INCLUDE
DBLIBDIR        = $(DBLIB)\LIB
ODBCLIBDIR       = $(ODBC)\LIB

TUX             = d:\tuxedo
TUXINCLUDE      = $(TUX)\INCLUDE
TUXLIB          = $(TUX)\LIB
TUXLIBS         = libtux.lib libbuft.lib libtux2.lib libfml.lib libfml32.lib
libgp.lib

!IF "$(CFG)" != "Debug"
LDEBUG       = /debug /pdb:none
CDEBUG       = /Zd /Yd
LDEBUG_RG    =
CDEBUG_RG    =
DEBUG        =
FLAGS        = /D "WIN32" /D "_WINDOWS"
OPT          = /Ot
!ELSE
LDEBUG       = /debug /pdb:$(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
$(TUXLIBS)
LINK32_OBJS1    = "$(OBJDIR)\tpcc1.obj" "$(OBJDIR)\tpcc1.res"
"$(OBJDIR)\tux_sql.obj" "$(OBJDIR)\tux_trans.obj" "$(OBJDIR)\error.obj"
"$(OBJDIR)\util.obj" "$(OBJDIR)\pipe_routines.obj" "$(OBJDIR)\tpcc_shm.obj"
LINK32_DEF1     = "$(SRCDIR)\tpcc1.def"
LINK32_FLAGS1   = /nologo /subsystem:windows /dll /incremental:no $(LDEBUG)
/def:"$(LINK32_DEF1)" /out:"$(OBJDIR)\tpcc.dll" /libpath:$(TUXLIB)

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)\tpcc.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"

"$(OBJDIR)\tux_trans.obj":   "$(SRCDIR)\tux_trans.c" "$(SRCDIR)\tpcc.h"
        cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(TUXINCLUDE) /I$(DBLIBINC)
$(FLAGS) /Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\tux_trans.obj /c
"$(SRCDIR)\tux_trans.c"

"$(OBJDIR)\tpcc_shm.obj":   "$(SRCDIR)\tpcc_shm.c" "$(SRCDIR)\tpcc_shm.h"
        cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(DBLIBINC) $(FLAGS)
/Fd$(OBJDIR)\tpcc1.pdb /Fo$(OBJDIR)\tpcc_shm.obj /c "$(SRCDIR)\tpcc_shm.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)\tpcc.dll:     $(LINK32_OBJS1) $(LINK32_DEF1)
        link.exe   $(LINK32_FLAGS1) $(LINK32_OBJS1) $(LINK32_LIBS1)
        rebase -v -b 0x67000000 -a -x $(OBJDIR)\Symbols $(OBJDIR)\tpcc.dll

"$(OBJDIR)\tpcc2.obj":   "$(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)\tpcc.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)\tpcc.dll
$(OBJDIR)\tpcc2.dll $(OBJDIR)\delisrv1.exe $(OBJDIR)\delisrv2.exe
$(OBJDIR)\install.res
        cl -W3 $(CDEBUG_RG) /Fo$(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)
<<
```

File: util.c

```
#include <windows.h>
#include <direct.h>
```

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


/* FUNCTION: void UtilStrCpy(char * pDest, char * pSrc, int n)
 *
 * PURPOSE:      This function copies n characters from string pSrc to pDst and
places a
 *                       null character at the end of the destination string.
 *
 * ARGUMENTS:    char                    *pDest   destination string pointer
 *                               char                    *pSrc    source
string pointer
 *                               int                             n
        number of characters to copy
 *
 * RETURNS:              None
 *
 * COMMENTS:     Unlike strncpy this function ensures that the result string is
 *                               always null terminated.
 *
 */

void UtilStrCpy(char * pDest, char * pSrc, int n)
{
    strncpy(pDest, pSrc, n);
    pDest[n] = '\0';

        return;
}

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

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

        int nBytes;

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

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

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

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

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

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

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

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

```
                                break;
                        }

                }
        }

        return bReturn;
}


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

        va_start(args, format);
        _strtime(buf);
        strcat(buf," ");
        len = strlen(buf);
        (void)_vsnprintf(buf+len, sizeof(buf)-len-1, format, args);

        va_end(args);

        fputs(buf, stderr);
}
```

File: util.h

```
#ifndef TPCC_UTIL_H
#define TPCC_UTIL_H

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

A.2 Driver

File: driver/generate.c

```
/*********************************************************************
 @(#) Version: A.10.10 $Date: 98/09/04 15:26:25 $

 (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;
    }
```

File: lib/random.h

```
/*****************************************************************************
 @(#) Version: A.10.10 $Date: 98/09/04 15:26:31 $

 (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

B.1 Create, backup and restore

CREATEDB.SQL
″

```
--  File:     CREATEDB.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates tpcc database and backup files


use master
go

--  remove any existing database and backup files

exec sp_dbremove tpcc, dropdev
exec sp_dropdevice 'tpccback1'
exec sp_dropdevice 'tpccback2'
exec sp_dropdevice 'tpccback3'
exec sp_dropdevice 'tpccback4'
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

--  create main database files

create database tpcc on
        (name="MSSQL70_tpcc_root",filename="D:\MSSQL70_tpcc_root.mdf",size=8MB,
FILEGROWTH=0)
log on
        (name="MSSQL70_tpcc_log",filename="E:",size=48000MB, FILEGROWTH=0)

--  create filegroups
```

```
alter database tpcc add filegroup MSSQL70_cs_fg
alter database tpcc add filegroup MSSQL70_misc_fg

--  add files to filegroups

alter database tpcc add file
        (name="MSSQL70_cs1",filename="F:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs2",filename="G:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs3",filename="H:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs4",filename="I:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs5",filename="J:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs6",filename="K:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs7",filename="L:",size=15900MB, FILEGROWTH=0),
        (name="MSSQL70_cs8",filename="M:",size=15900MB, FILEGROWTH=0)
to filegroup MSSQL70_cs_fg

alter database tpcc add file
        (name="MSSQL70_misc1",filename="N:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc2",filename="O:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc3",filename="P:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc4",filename="Q:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc5",filename="R:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc6",filename="S:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc7",filename="T:",size=8000MB, FILEGROWTH=0),
        (name="MSSQL70_misc8",filename="U:",size=8000MB, FILEGROWTH=0)
to filegroup MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)
go

--  create backup devices

exec sp_addumpdevice 'disk','tpccback1','x:\tpccback1.dmp'
exec sp_addumpdevice 'disk','tpccback2','x:\tpccback2.dmp'
exec sp_addumpdevice 'disk','tpccback3','x:\tpccback3.dmp'
exec sp_addumpdevice 'disk','tpccback4','x:\tpccback4.dmp'
go
```

BACKUP.SQL

```
--  File:     BACKUP.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates backup of tpcc database


declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

dump database tpcc to tpccback1, tpccback2, tpccback3, tpccback4 with init,
stats = 1

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

RESTORE.SQL

```
--  File:     RESTORE.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Loads database backup from backup files


declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

load database tpcc from tpccback1, tpccback2, tpccback3, tpccback4 with stats =
1, replace

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

B.2 Build indices

IDXCUSCL.SQL
˝

```
--  File:     IDXCUSCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on customer table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index customer_c1 on customer(c_w_id, c_d_id, c_id)
        on MSSQL70_cs_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXCUSNC.SQL

```
--  File:     IDXCUSNC.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates non-clustered index on customer table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique nonclustered index customer_nc1 on customer(c_w_id, c_d_id,
c_last, c_first, c_id)
        on MSSQL70_cs_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXDISCL.SQL

```
--  File:     IDXDISCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on district table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where name = 'district_c1' )
```

```
        drop index district.district_c1

create unique clustered index  district_c1 on district(d_w_id, d_id)
        with fillfactor=100 on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXITMCL.SQL

```
--  File:     IDXITMCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on item table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index item_c1 on item(i_id)
        on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXNODCL.SQL

```
--  File:     IDXNODCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on new_order table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index new_order_c1 on new_order(no_w_id, no_d_id,
no_o_id)
        on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXODLCL.SQL

```
--  File:     IDXODLCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on order_line table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
```

```
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index order_line_c1 on order_line(ol_w_id, ol_d_id,
ol_o_id, ol_number)
        on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXORDCL.SQL

```
--  File:     IDXORDCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on orders table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index orders_c1 on orders(o_w_id, o_d_id, o_id)
        on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXORDNC.SQL

```
--  File:     IDXORDNC.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates non-clustered index on orders table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes where  name = 'orders_nc1' )
        drop index orders.orders_nc1

create index orders_nc1 on orders(o_w_id, o_d_id, o_c_id, o_id)
        on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXSTKCL.SQL

```
--  File:     IDXSTKCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on stock table


use tpcc
go

declare @startdate datetime
```

```
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index stock_c1 on stock(s_i_id, s_w_id)
        on MSSQL70_cs_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)

go
```

IDXWARCL.SQL

```
--  File:     IDXWARCL.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates clustered index on warehouse table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:", convert(varchar(30),@startdate,9)

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

create unique clustered index warehouse_c1 on warehouse(w_id)
        with fillfactor=100 on MSSQL70_misc_fg

select @enddate = getdate()
select "End date: ", convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ", datediff(second, @startdate, @enddate)
go
```

B.3 Database Options

DBOPT1.SQL
"

```
--  File:     DBOPT1.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Sets database options for data load


use master
go

exec sp_dboption tpcc,'select into/bulkcopy',true
exec sp_dboption tpcc,'trunc. log on chkpt.',true
go

use tpcc
go

checkpoint
go
```

DBOPT2.SQL

```
--  File:     DBOPT2.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.01
--            Copyright Microsoft, 1996
--  Purpose:  Resets database options after data load


use master
go
```

```
sp_dboption tpcc,'select ',false
go

sp_dboption tpcc,'trunc. ',false
go

use tpcc
go

checkpoint
go

sp_configure allow,1
go

reconfigure with override
go

/*                                          */
/* Set option values for user-defined indexes */
/*                                          */

sp_indexoption 'customer','AllowPageLocks',FALSE
go
sp_indexoption 'district','AllowPageLocks',FALSE
go
sp_indexoption 'warehouse','AllowPageLocks',FALSE
go
sp_indexoption 'stock','AllowPageLocks',FALSE
go
sp_indexoption 'order_line','AllowRowLocks',FALSE
go
sp_indexoption 'orders','AllowRowLocks',FALSE
go
sp_indexoption 'new_order','AllowRowLocks',FALSE
go
sp_indexoption 'item','AllowRowLocks',FALSE
go
sp_indexoption 'item','AllowPageLocks',FALSE
go

Print ' '
Print '******************'
Print 'Pre-specified Locking Hierarchy:'
Print '   Lockflag = 0 ==> No pre-specified hierarchy'
Print '   Lockflag = 1 ==> Lock at Page-level then Table-level'
Print '   Lockflag = 2 ==> Lock at Row-level then Table-level'
Print '   Lockflag = 3 ==> Lock at Table-level'
Print ' '

select name,lockflags
from sysindexes
where object_id("warehouse")=id or
          object_id("district")=id or
          object_id("customer")=id or
          object_id("stock")=id or
          object_id("orders")=id or
          object_id("order_line")=id or
          object_id("history")=id or
          object_id("new_order")=id or
          object_id("item")=id
order by lockflags asc
go

sp_configure allow,0
go

reconfigure with override
go

exec sp_dboption tpcc, 'auto update statistics', FALSE
exec sp_dboption tpcc, 'auto create statistics', FALSE
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
```

B.4 Table definitions

TABLES.SQL

```
--  File:     TABLES.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates TPC-C tables


use tpcc
go

if exists ( select name from sysobjects where name = 'warehouse' )
        drop table warehouse
go
create table warehouse
(
        w_id                            smallint,
        w_name                          char(10),
        w_street_1                      char(20),
        w_street_2                      char(20),
        w_city                          char(20),
        w_state                         char(2),
        w_zip                           char(9),
        w_tax                           numeric(4,4),
        w_ytd                           numeric(12,2)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'district' )
        drop table district
go
create table district
(
        d_id                            tinyint,
        d_w_id                          smallint,
        d_name                          char(10),
        d_street_1                      char(20),
        d_street_2                      char(20),
        d_city                          char(20),
        d_state                         char(2),
        d_zip                           char(9),
        d_tax                           numeric(4,4),
        d_ytd                           numeric(12,2),
        d_next_o_id                     int
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'customer' )
        drop table customer
go
create table customer
(
        c_id                                    int,
        c_d_id                                  tinyint,
        c_w_id                                  smallint,
        c_first                                 char(16),
        c_middle                        char(2),
        c_last                                  char(16),
        c_street_1                              char(20),
        c_street_2                              char(20),
        c_city                                  char(20),
        c_state                                 char(2),
        c_zip                                   char(9),
        c_phone                                 char(16),
        c_since                                 datetime,
        c_credit                        char(2),
        c_credit_lim                    numeric(12,2),
        c_discount                              numeric(4,4),
        c_balance                               numeric(12,2),
        c_ytd_payment                   numeric(12,2),
        c_payment_cnt                   smallint,
        c_delivery_cnt                  smallint,
        c_data                                  char(500)
) on MSSQL70_cs_fg
go

if exists ( select name from sysobjects where name = 'history' )
        drop table history
go
create table history
(
        h_c_id                                  int,
        h_c_d_id                        tinyint,
        h_c_w_id                        smallint,
        h_d_id                                  tinyint,
        h_w_id                                  smallint,
        h_date                                  datetime,
        h_amount                        numeric(6,2),
        h_data                                  char(24)
```

```
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'new_order' )
        drop table new_order
go
create table new_order
(
        no_o_id                         int,
        no_d_id                         tinyint,
        no_w_id                         smallint
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'orders' )
        drop table orders
go
create table orders
(
        o_id                            int,
        o_d_id                          tinyint,
        o_w_id                          smallint,
        o_c_id                          int,
        o_entry_d                       datetime,
        o_carrier_id            tinyint,
        o_ol_cnt                tinyint,
        o_all_local                     tinyint
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'order_line' )
        drop table order_line
go
create table order_line
(
        ol_o_id                         int,
        ol_d_id                         tinyint,
        ol_w_id                         smallint,
        ol_number                       tinyint,
        ol_i_id                         int,
        ol_supply_w_id          smallint,
        ol_delivery_d           datetime,
        ol_quantity                     smallint,
        ol_amount                       numeric(6,2),
        ol_dist_info            char(24)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'item' )
        drop table item
go
create table item
(
        i_id                            int,
        i_im_id                         int,
        i_name                          char(24),
        i_price                         numeric(5,2),
        i_data                          char(50)
) on MSSQL70_misc_fg
go

if exists ( select name from sysobjects where name = 'stock' )
        drop table stock
go
create table stock
(
        s_i_id                          int,
        s_w_id                          smallint,
        s_quantity                      smallint,
        s_dist_01                       char(24),
        s_dist_02                       char(24),
        s_dist_03                       char(24),
        s_dist_04                       char(24),
        s_dist_05                       char(24),
        s_dist_06                       char(24),
        s_dist_07                       char(24),
        s_dist_08                       char(24),
        s_dist_09                       char(24),
        s_dist_10                       char(24),
        s_ytd                           int,
        s_order_cnt                     smallint,
        s_remote_cnt            smallint,
        s_data                          char(50)
) on MSSQL70_cs_fg
go
```

B.5 Stored Procedures

DELIVERY.SQL

```
--  File:     DELIVERY.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates delivery transaction stored procedure


use tpcc
go

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 top 1 @o_id = no_o_id
          from new_order (serializable updlock)
    where no_w_id = @w_id and
                 no_d_id = @d_id
          order by no_o_id asc

    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

-- return delivery data to client

select @oid1,
       @oid2,
       @oid3,
       @oid4,
       @oid5,
       @oid6,
       @oid7,
       @oid8,
       @oid9,
       @oid10

go
```

NEWORD.SQL

```
--  File:     NEWORD.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates new order transaction stored procedure
--
--  Modified 6/24/98 - Jamie Reding - Microsoft Corporation
--       Replaced Select of @s_quantity with Select of @li_qty to insure
--       correct data when sending line-item data to client.
--

use tpcc
go

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

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,
```

```
@s_w_id12 smallint = 0, @ol_qty12 smallint = 0,

@s_w_id13 smallint = 0, @ol_qty13 smallint = 0,

@s_w_id14 smallint = 0, @ol_qty14 smallint = 0,

@s_w_id15 smallint = 0, @ol_qty15 smallint = 0


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

begin

        begin transaction n

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

    update district
        set @d_tax      = d_tax,
            @o_id       = d_next_o_id,
                d_next_o_id = d_next_o_id + 1,
@i_id12 int = 0,

@i_id13 int = 0,

@i_id14 int = 0,

@i_id15 int = 0,
                @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 repeatableread)
                 where i_id = @li_id

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

                 if (@@rowcount > 0)
                 begin
                 select @s_dist = NULL
           update stock set s_ytd        = s_ytd + @li_qty,
                        @s_quantity  = s_quantity =
                        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,
                                               @d_id,
                                               @w_id,
                                               @li_no,
                                               @li_id,
                                               @li_s_w_id,
                                               "dec 31, 1899",
                                               @li_qty,
                                               @i_price * @li_qty,
                                               @s_dist)


--  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 (repeatableread)
    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 (repeatableread)
        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
```

ORDSTAT.SQL

```
--  File:     ORDSTAT.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates order status transaction stored procedure


use tpcc
go

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

create proc tpcc_orderstatus @w_id                smallint,
                                                          @d_id
        tinyint,
                                                          @c_id          int,
```

```
@c_last char(16) =
""

as

declare @c_balance              numeric(12,2),
        @c_first        char(16),
        @c_middle               char(2),
        @o_id                   int,
        @o_entry_d              datetime,
        @o_carrier_id   smallint,
        @cnt                    smallint

begin tran o

        if (@c_id = 0)
                begin

--  get customer id and info using last name

                select @cnt = (count(*)+1)/2
                from customer (repeatableread)
                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 (repeatableread)
                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 (repeatableread)
                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 (serializable)
        where o_c_id = @c_id and
          o_d_id = @d_id and
          o_w_id = @w_id
        order by o_id asc

--  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 (repeatableread)
        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
```

PAYMENT.SQL

```
--  File:     PAYMENT.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates payment transaction stored procedure


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),
         @data           char(500),
         @c_data         char(500),
         @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 (repeatableread)
                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 (repeatableread)
                where c_last = @c_last and
                      c_w_id = @c_w_id and
                      c_d_id = @c_d_id
                order by 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,
                @data          = c_data,
                @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 = 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(@data, 1, 458)

--  update customer info

                update customer set
                       c_data = @c_data
                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,1,200)
        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
```

STOCKLEV.SQL

```
--  File:     STOCKLEV.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Creates stock level transaction stored procedure


use tpcc
go

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

create proc tpcc_stocklevel      @w_id           smallint,
                                 @d_id           tinyint,
```

```
                                                    @threshhold      smallint
as

        declare @o_id_low int,
                    @o_id_high int

    select @o_id_low  = (d_next_o_id - 20),
               @o_id_high = (d_next_o_id - 1)
    from district
    where d_w_id = @w_id and
                  d_id   = @d_id

    select count(distinct(s_i_id))
        from  stock, order_line
    where ol_w_id    =  @w_id and
              ol_d_id    =  @d_id and
              ol_o_id between @o_id_low and @o_id_high and
          s_w_id     =  ol_w_id and
              s_i_id     =  ol_i_id and
              s_quantity <  @threshhold

go
```

Appendix C Tunable Parameters

C.1 Server Windows NT Configuration Parameters

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

- Alerter
- Computer Browser
- Gopher Publishing Service
- FTP Publishing Service
- License Logging Service
- TCP/IP NetBIOS Helper
- Messenger
- NT LM Security Support Provider
- Plug and Play
- Spooler
- World Wide Web Publishing Service
- schedule

Below is the boot.ini file.

[boot loader]
timeout=10
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server, Enterprise Edition Version 4.00" /3GB /SOS /maxmem=2816
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Server, Enterprise Edition Version 4.00 [VGA mode]" /basevideo /sos
C:\="MS-DOS"

C.2 Server System Configuration Parameters

```
Microsoft Diagnostics Report For \\PRF_SUT7
------------------------------------------------------------------------

OS Version Report
------------------------------------------------------------------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiprocessor Free
Registered Owner: Hewlett-Packard, TPCC-Lab
Product Number: 70234-810-3897177-09592
------------------------------------------------------------------------

System Report
------------------------------------------------------------------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 11/20/98
BIOS Version: AC450NX - PhoenixBIOS 4.0 Releas

Processor list:
   0:  x86 Family 6 Model 5 Stepping 3 GenuineIntel ~450 Mhz
   1:  x86 Family 6 Model 5 Stepping 3 GenuineIntel ~450 Mhz
   2:  x86 Family 6 Model 5 Stepping 3 GenuineIntel ~450 Mhz
   3:  x86 Family 6 Model 5 Stepping 3 GenuineIntel ~450 Mhz
------------------------------------------------------------------------

Video Display Report
------------------------------------------------------------------------
BIOS Date: 05/29/98
BIOS Version: CL-GD5446 PCI VGA BIOS Version 1.35

Adapter:
   Setting: 1024 x 768 x 256
            60 Hz
   Type: cirrus compatible display adapter
   String: Cirrus Logic Compatible
   Memory: 2 MB
   Chip Type: Cirrus Logic 5446
   DAC Type: Integrated RAMDAC
Driver:
```

```
   Vendor: Microsoft Corporation
   File(s): cirrus.sys, vga.dll, cirrus.dll, vga256.dll, vga64K.dll
   Version: 4.00, 4.0.0


Drives Report
-----------------------------------------------------------------------
C:\  (Local - FAT)  Total: 2,096,160KB, Free: 1,373,088KB
   Serial Number: 3B6B - 12F4
   Bytes per cluster: 512
   Sectors per cluster: 64
   Filename length: 255
D:\  (Local - NTFS)  Total: 6,787,460KB, Free: 2,581,452KB
   Serial Number: 5C76 - 933D
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
V:  (Remote - NTFS) \\nrddata\g$  Total: 44,431,036KB, Free: 13,525,472KB
   Serial Number: C0EF - 452A
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
W:\  (Local - NTFS)  Total: 182,525,948KB, Free: 37,706,756KB
   Serial Number: 584D - D1F2
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
X:\  (Local - NTFS)  Total: 182,525,948KB, Free: 36,534,216KB
   Serial Number: 982A - 509F
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255
Y:\  (Local - NTFS)  Total: 182,525,948KB, Free: 39,253,512KB
   Serial Number: D034 - A706
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255

Memory Report
-----------------------------------------------------------------------
Handles: 2,285
Threads: 168
Processes: 20

Physical Memory (K)
   Total: 2,882,952
   Available: 959,152
   File Cache: 28,968

Kernel Memory (K)
   Total: 1,086,564
   Paged: 8,472
   Nonpaged: 1,078,092

Commit Charge (K)
   Total: 1,814,096
   Limit: 6,960,768
   Peak: 1,814,188

Pagefile Space (K)
   Total: 4,193,280
   Total in use: 3,204
   Peak: 3,244

   D:\pagefile.sys
      Total: 4,193,280
      Total in use: 3,204
      Peak: 3,244


Services Report
-----------------------------------------------------------------------
Alerter                                      Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
Ataman TCP Remote Logon Services             Running   (Automatic)
   C:\ATRLS2\ATRLS.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Computer Browser                             Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
```

```
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
      LmHosts
ClipBook Server                                  Stopped   (Manual)
   C:\WINNT\system32\clipsrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      NetDDE
DHCP Client (TDI)                                Stopped   (Disabled)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Tcpip
      Afd
      NetBT
EventLog (Event log)                             Running   (Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Gopher Publishing Service                        Running   (Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
Server                                           Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
Workstation (NetworkProvider)                    Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
License Logging Service                          Running   (Automatic)
   C:\WINNT\System32\llssrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TCP/IP NetBIOS Helper                            Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      NetworkProvider
Messenger                                        Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      NetBios
Monitor Service                                  Stopped   (Manual)
   C:\WINNT\system32\datalog.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
MSDTC (MS Transactions)                          Stopped   (Manual)
   C:\WINNT\System32\msdtc.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      RPCSS
      NTLMSSP
FTP Publishing Service                           Running   (Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
```

```
      RPCSS
      NTLMSSP
MSSQLServer                                  Stopped   (Manual)
   C:\MSSQL7\binn\sqlservr.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Network DDE (NetDDEGroup)                    Stopped   (Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      NetDDEDSDM
Network DDE DSDM                             Stopped   (Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Net Logon (RemoteValidation)                 Stopped   (Manual)
   C:\WINNT\System32\lsass.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LmHosts
NT LM Security Support Provider              Running   (Manual)
   C:\WINNT\System32\SERVICES.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Plug and Play (PlugPlay)                     Running   (Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Protected Storage                            Running   (Automatic)
   C:\WINNT\System32\pstores.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
   Service Dependencies:
      RpcSs
Directory Replicator                         Stopped   (Manual)
   C:\WINNT\System32\lmrepl.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
Remote Procedure Call (RPC) Locator          Stopped   (Manual)
   C:\WINNT\System32\LOCATOR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      Rdr
Remote Procedure Call (RPC) Service          Running   (Automatic)
   C:\WINNT\system32\RpcSs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Schedule                                     Stopped   (Manual)
   C:\WINNT\System32\AtSvc.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Spooler (SpoolerGroup)                       Running   (Automatic)
   C:\WINNT\system32\spoolss.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
SQLServerAgent                               Stopped   (Manual)
   C:\MSSQL7\binn\sqlagent.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      MSSQLServer
Telephony Service                            Stopped   (Manual)
   C:\WINNT\system32\tapisrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
UPS                                          Stopped   (Manual)
```

```
   C:\WINNT\System32\ups.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
World Wide Web Publishing Service             Running   (Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP



Drivers Report
--------------------------------------------------------------------
Abiosdsk (Primary disk)                       Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
AFD Networking Support Environment (TDI)      Running   (Automatic)
   C:\WINNT\System32\drivers\afd.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha154x (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha174x (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
aic78xx (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Always (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ami0nt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
amsint (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Arrow (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
atapi (SCSI miniport)                         Running   (Boot)
   C:\WINNT\System32\DRIVERS\atapi.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Atdisk (Primary disk)                         Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ati (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Beep (Base)                                   Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
BusLogic (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Busmouse (Pointer Port)                       Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdaudio (Filter)                              Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdfs (File system)                            Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      SCSI CDROM Class
Cdrom (SCSI CDROM Class)                      Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Changer (Filter)                              Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
cirrus (Video)                                Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Cpqarray (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
cpqfws2e (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
```

```
   Service Flags: Kernel Driver, Shared Process
dac960nt (SCSI miniport)                    Stopped   (Boot)
   C:\WINNT\System32\drivers\dac960nt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dce376nt (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Delldsa (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Dell_DGX (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Disk (SCSI Class)                           Running   (Boot)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Diskperf (Filter)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
DptScsi (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dtc329x (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Intel 82557-based PRO Adapter Driver (NDIS) Running   (Automatic)
   C:\WINNT\System32\drivers\e100b.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
em (Base)                                   Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
et4000 (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Fastfat (Boot file system)                  Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Fd16_700 (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd7000ex (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd8xx (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
flashpnt (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Floppy (Primary disk)                       Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ftdisk (Filter)                             Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
gamdrv (SCSI Class)                         Stopped   (Boot)
   C:\WINNT\System32\drivers\gamdrv.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running   (System)
   System32\DRIVERS\i8042prt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Inport (Pointer Port)                       Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg300 (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg364 (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jzvxl484 (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Keyboard Class Driver (Keyboard Class)      Running   (System)
   System32\DRIVERS\kbdclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
KSecDD (Base)                               Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
macdisk (Filter)                            Stopped   (Boot)
   C:\WINNT\System32\drivers\macdisk.sys
```

```
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mga (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mga_mil (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mitsumi (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mkecr5xx (SCSI miniport)                   Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Modem (Extended base)                      Stopped   (Manual)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Mouse Class Driver (Pointer Class)         Running   (System)
   System32\DRIVERS\mouclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mraid (Primary disk)                       Running   (Boot)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Msfs (File system)                         Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                              Running   (Manual)
   C:\WINNT\System32\drivers\mup.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ncr53c9x (SCSI miniport)                   Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ncr77c22 (Video)                           Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ncrc700 (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ncrc710 (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Microsoft NDIS System Driver (NDIS)        Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
NetBIOS Interface (NetBIOSGroup)           Running   (Manual)
   C:\WINNT\System32\drivers\netbios.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      TDI
WINS Client(TCP/IP) (PNP_TDI)              Running   (Automatic)
   C:\WINNT\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                                  Stopped   (Manual)
   C:\WINNT\system32\drivers\netdtect.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Npfs (File system)                         Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ntfs (File system)                         Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Null (Base)                                Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Oliscsi (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Parallel (Extended base)                   Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
Parport (Parallel arbitrator)              Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ParVdm (Extended base)                     Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
```

```
      Parport
   Group Dependencies:
      Parallel arbitrator
PCIDump (PCI Configuration)                    Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Pcmcia (System Bus Extender)                   Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
PnP ISA Enabler Driver (Base)                  Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
psidisp (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
PStat                                          Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ql10wnt (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
qv (Video)                                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Rdr (Network)                                  Running   (Manual)
   C:\WINNT\System32\drivers\rdr.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
s3 (Video)                                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Scsiprnt (Extended base)                       Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Scsiscan (SCSI Class)                          Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Serial (Extended base)                         Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sermouse (Pointer Port)                        Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sfloppy (Primary disk)                         Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Simbad (Filter)                                Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
slcd32 (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Sparrow (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Spock (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Srv (Network)                                  Running   (Manual)
   C:\WINNT\System32\drivers\srv.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
symc810 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
sym_hi (SCSI miniport)                         Running   (Boot)
   C:\WINNT\system32\drivers\sym_hi.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T128 (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T13B (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
TCP/IP Service (PNP_TDI)                       Running   (Automatic)
   C:\WINNT\System32\drivers\tcpip.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
tga (Video)                                    Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
```

```
tmv1 (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra124 (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra14f (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra24f (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
v7vram (Video)                                Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaSave (Video Save)                          Stopped   (System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                         Stopped   (System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Wd33c93 (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
wd90c24a (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
wdvga (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
weitekp9 (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Xga (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process



IRQ and Port Report
-------------------------------------------------------------------------
Devices                        Vector Level  Affinity
```

```
-------------------------------------------------------------------------
MPS 1.4 - APIC platform           8     8 0x0000000f
MPS 1.4 - APIC platform           0     0 0x0000000f
MPS 1.4 - APIC platform           1     1 0x0000000f
MPS 1.4 - APIC platform           2     2 0x0000000f
MPS 1.4 - APIC platform           3     3 0x0000000f
MPS 1.4 - APIC platform           4     4 0x0000000f
MPS 1.4 - APIC platform           5     5 0x0000000f
MPS 1.4 - APIC platform           6     6 0x0000000f
MPS 1.4 - APIC platform           7     7 0x0000000f
MPS 1.4 - APIC platform           8     8 0x0000000f
MPS 1.4 - APIC platform           9     9 0x0000000f
MPS 1.4 - APIC platform          10    10 0x0000000f
MPS 1.4 - APIC platform          11    11 0x0000000f
MPS 1.4 - APIC platform          12    12 0x0000000f
MPS 1.4 - APIC platform          13    13 0x0000000f
MPS 1.4 - APIC platform          14    14 0x0000000f
MPS 1.4 - APIC platform          15    15 0x0000000f
MPS 1.4 - APIC platform          16    16 0x0000000f
MPS 1.4 - APIC platform          17    17 0x0000000f
MPS 1.4 - APIC platform          18    18 0x0000000f
MPS 1.4 - APIC platform          19    19 0x0000000f
MPS 1.4 - APIC platform          20    20 0x0000000f
MPS 1.4 - APIC platform          21    21 0x0000000f
MPS 1.4 - APIC platform          22    22 0x0000000f
MPS 1.4 - APIC platform          23    23 0x0000000f
MPS 1.4 - APIC platform          24    24 0x0000000f
MPS 1.4 - APIC platform          25    25 0x0000000f
MPS 1.4 - APIC platform          26    26 0x0000000f
MPS 1.4 - APIC platform          27    27 0x0000000f
MPS 1.4 - APIC platform          28    28 0x0000000f
MPS 1.4 - APIC platform          29    29 0x0000000f
MPS 1.4 - APIC platform          30    30 0x0000000f
MPS 1.4 - APIC platform          31    31 0x0000000f
MPS 1.4 - APIC platform          32    32 0x0000000f
MPS 1.4 - APIC platform          33    33 0x0000000f
MPS 1.4 - APIC platform          34    34 0x0000000f
MPS 1.4 - APIC platform          35    35 0x0000000f
MPS 1.4 - APIC platform          36    36 0x0000000f
MPS 1.4 - APIC platform          37    37 0x0000000f
MPS 1.4 - APIC platform          38    38 0x0000000f
MPS 1.4 - APIC platform          39    39 0x0000000f
MPS 1.4 - APIC platform          40    40 0x0000000f
MPS 1.4 - APIC platform          41    41 0x0000000f
```

```
MPS 1.4 - APIC platform          42    42 0x0000000f
MPS 1.4 - APIC platform          43    43 0x0000000f
MPS 1.4 - APIC platform          44    44 0x0000000f
MPS 1.4 - APIC platform          45    45 0x0000000f
MPS 1.4 - APIC platform          46    46 0x0000000f
MPS 1.4 - APIC platform          47    47 0x0000000f
MPS 1.4 - APIC platform          61    61 0x0000000f
MPS 1.4 - APIC platform          65    65 0x0000000f
MPS 1.4 - APIC platform          80    80 0x0000000f
MPS 1.4 - APIC platform         193   193 0x0000000f
MPS 1.4 - APIC platform         225   225 0x0000000f
MPS 1.4 - APIC platform         253   253 0x0000000f
MPS 1.4 - APIC platform         254   254 0x0000000f
MPS 1.4 - APIC platform         255   255 0x0000000f
i8042prt                          1     1 0xffffffff
i8042prt                         12    12 0xffffffff
Serial                            4     4 0x00000000
Serial                            3     3 0x00000000
E100B                            36    36 0x6db6db6d
Floppy                            6     6 0x00000000
atapi                             0    14 0x00000000
sym_hi                           40    40 0x00000000
sym_hi                           41    41 0x00000000
------------------------------------------------------------------------
Devices                       Physical Address  Length
------------------------------------------------------------------------
MPS 1.4 - APIC platform         0x00000000  0x0000000010
MPS 1.4 - APIC platform         0x00000020  0x0000000002
MPS 1.4 - APIC platform         0x00000040  0x0000000004
MPS 1.4 - APIC platform         0x00000048  0x0000000004
MPS 1.4 - APIC platform         0x00000061  0x0000000001
MPS 1.4 - APIC platform         0x00000070  0x0000000002
MPS 1.4 - APIC platform         0x00000080  0x0000000010
MPS 1.4 - APIC platform         0x00000092  0x0000000001
MPS 1.4 - APIC platform         0x000000a0  0x0000000002
MPS 1.4 - APIC platform         0x000000c0  0x0000000010
MPS 1.4 - APIC platform         0x000000f0  0x0000000010
i8042prt                        0x00000060  0x0000000001
i8042prt                        0x00000064  0x0000000001
Parport                         0x00000378  0x0000000003
Serial                          0x000003f8  0x0000000007
Serial                          0x000002f8  0x0000000007
E100B                           0x00002800  0x0000000014
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
atapi                           0x000001f0  0x0000000008
atapi                           0x000003f6  0x0000000001
sym_hi                          0x00002000  0x0000000100
sym_hi                          0x00002400  0x0000000100
cirrus                          0x000003b0  0x000000000c
cirrus                          0x000003c0  0x0000000020


DMA and Memory Report
------------------------------------------------------------------------
Devices                       Channel    Port
------------------------------------------------------------------------
Floppy                           2     0
------------------------------------------------------------------------
Devices                       Physical Address  Length
------------------------------------------------------------------------
MPS 1.4 - APIC platform         0xfec10000  0x00000400
MPS 1.4 - APIC platform         0xfee00000  0x00000400
E100B                           0xfa400000  0x00000014
E100B                           0xfa400000  0x00000014
sym_hi                          0xfa004000  0x00000400
sym_hi                          0xfa000000  0x00002000
sym_hi                          0xfa004400  0x00000400
sym_hi                          0xfa002000  0x00002000
cirrus                          0x000a0000  0x00020000
cirrus                          0xfb000000  0x01000000


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


System Environment Variables
   ComSpec=C:\WINNT\system32\cmd.exe
   HOME=C:/
   NTRESKIT=C:\NTRESKIT
   NUMBER_OF_PROCESSORS=4
   OS=Windows_NT
   Os2LibPath=C:\WINNT\system32\os2\dll;
```

```
Path=C:\mksnt;C:\WINNT\system32;C:\WINNT;C:\NTRESKIT;C:\NTRESKIT\Perl;C:\MSSQL7
\BINN
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 5 Stepping 3, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=0503
   ROOTDIR=C:/
   SHELL=C:/mksnt/sh.exe
   TMPDIR=C:/TEMP
   windir=C:\WINNT



Environment Variables for Current User
   TEMP=C:\TEMP
   TMP=C:\TEMP



Network Report
------------------------------------------------------------------------
Your Access Level: Admin & Local
Workgroup or Domain: WORKGROUP
Network Version: 4.0
LanRoot: WORKGROUP
Logged On Users: 1
Current User (1): Administrator
  Logon Domain: PRF_SUT7
  Logon Server: PRF_SUT7

Transport: NetBT_E100B1, 00-A0-C9-B1-48-F6, VC's: 1, Wan: Wan

Character Wait: 3,600
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 17
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
Log Election Packets: False
Use Opportunistic Locking: True
Use Unlock Behind: True
Use Close Behind: True
Buffer Pipes: True
Use Lock, Read, Unlock: True
Use NT Caching: True
Use Raw Read: True
Use Raw Write: True
Use Write Raw Data: True
Use Encryption: True
Buffer Deny Write Files: True
Buffer Read Only Files: True
Force Core Creation: True
512 Byte Max Transfer: False
Bytes Received: 12,059
SMB's Received: 79
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Bytes Transmitted: 8,146
SMB's Transmitted: 79
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 420
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Initally Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 0
Random Read Operations: 0
Read SMB's: 0
Large Read SMB's: 0
Small Read SMB's: 0
Write Operations: 7
```

```
Random Write Operations: 0
Write SMB's: 0
Large Write SMB's: 0
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 2
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 1
Server Disconnects: 0
Hung Sessions: 0
Use Count: 2
Failed Use Count: 0
Current Commands: 0
Server File Opens: 0
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 0
Server Sessions Timed Out: 0
Server Sessions Errored Out: 0
Server Password Errors: 0
Server Permission Errors: 0
Server System Errors: 0
Server Bytes Sent: 0
Server Bytes Received: 0
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

C.3 Microsoft SQL Server 7.0 Startup Parameters

```
sqlservr -x -c -T3502
```

C.4 Microsoft SQL Server 7.0 Stack Size

The default stack size for Microsoft SQL Server was changed using the EDITBIN utility, which ships with Microsoft Visual C++. The command used to change the stack size is:

```
editbin /Stack:131072 sqlservr.exe
```

This command is fully documented as an article in the Microsoft Knowledge Base on the Microsoft Web Site at www.microsoft.com/support

C.5 Microsoft SQL Server 7.0 Configuration Parameters

name	minimum	maximum	config_value	run_value
affinity mask	0	2147483647	15	15
allow updates	0	1	0	0
cost threshold for parallelism	0	32767	5	5
cursor threshold	-1	2147483647	-1	-1
default language	0	9999	0	0
default sortorder id	0	255	50	50
extended memory size (MB)	0	2147483647	0	0
fill factor (%)	0	100	0	0
index create memory (KB)	704	1600000	0	0
language in cache	3	100	3	3
lightweight pooling	0	1	1	1
locks	5000	2147483647	0	0
max async IO	1	255	255	255
max degree of parallelism	0	32	1	1
max server memory (MB)	4	2147483647	2800	2800
max text repl size (B)	0	2147483647	65536	65536
max worker threads	10	1024	200	200
media retention	0	365	0	0
min memory per query (KB)	512	2147483647	1024	1024
min server memory (MB)	0	2147483647	0	0
nested triggers	0	1	1	1
network packet size (B)	512	65535	4096	4096
open objects	0	2147483647	0	0
priority boost	0	1	1	1
query governor cost limit	0	2147483647	0	0
query wait (s)	-1	2147483647	-1	-1
recovery interval (min)	0	32767	32767	32767
remote access	0	1	1	1
remote login timeout (s)	0	2147483647	30	30
remote proc trans	0	1	0	0
remote query timeout (s)	0	2147483647	0	0
resource timeout (s)	5	2147483647	10	10
scan for startup procs	0	1	0	0
set working set size	0	1	0	0
show advanced options	0	1	1	1
spin counter	1	2147483647	10000	10000
time slice (ms)	50	1000	100	100
Unicode comparison style	0	2147483647	0	0
Unicode locale id	0	2147483647	33280	33280
user connections	0	32767	300	300
user options	0	4095	0	0

(41 row(s) affected)

C.6 Internal DAC Configuration Parameters

The AMI Series 438-H Super High Performance RAID Controller DACs have 3 channels, each of them connected to 8 physical drives. The drives are stripped across the three channels so that the 24 drives in each DAC look as one logical drive to NT. The drives are configured in Raid 0 in Write Thru mode on the controller.

C.7 External DAC Configuration Parameters

```
    ************************************************************************
″
    *        MYLEX Disk Array Controller - Configuration Utility           *
    *                            Version 7.09                              *
    ************************************************************************

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

    5 Channel - 16 Target  DAC960SX  Firmware version 3.24-00

    Daughter Card


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

    Number of Packs = 1

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

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

    Number of System Drives = 1
    LUN Affinity = Simple

    # Phy. Size  Raid  Eff. Size  Cache       Packs  Status
    = =========  ====  =========  ==========  =====  ======
    0 138846 MB    1    69423 MB  Write Thru  A...   Onln

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

    Simple mapping:
    #     c0p0   c0p1   c1p0   c1p1
    =     ====   ====   ====   ====
    0      0      0      .      .

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

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

```
2    2   Absent    00   DEAD          0    00000000
2    3   Absent    00   DEAD          0    00000000
2    4   Absent    00   DEAD          0    00000000
2    5   Absent    00   DEAD          0    00000000
2    6   Absent    00   DEAD          0    00000000
2    8   Absent    00   DEAD          0    00000000
2    9   Absent    00   DEAD          0    00000000
2    a   Absent    00   DEAD          0    00000000
2    b   Absent    00   DEAD          0    00000000
2    c   Absent    00   DEAD          0    00000000
2    d   Absent    00   DEAD          0    00000000
2    e   Absent    00   DEAD          0    00000000
2    f   Absent    00   DEAD          0    00000000
3    0   Present   f1   ONLINE        0    00000000
3    1   Absent    00   DEAD          0    00000000
3    2   Absent    00   DEAD          0    00000000
3    3   Absent    00   DEAD          0    00000000
3    4   Absent    00   DEAD          0    00000000
3    5   Absent    00   DEAD          0    00000000
3    6   Absent    00   DEAD          0    00000000
3    8   Absent    00   DEAD          0    00000000
3    9   Absent    00   DEAD          0    00000000
3    a   Absent    00   DEAD          0    00000000
3    b   Absent    00   DEAD          0    00000000
3    c   Absent    00   DEAD          0    00000000
3    d   Absent    00   DEAD          0    00000000
3    e   Absent    00   DEAD          0    00000000
3    f   Absent    00   DEAD          0    00000000
4    0   Absent    00   DEAD          0    00000000
4    1   Absent    00   DEAD          0    00000000
4    2   Absent    00   DEAD          0    00000000
4    3   Absent    00   DEAD          0    00000000
4    4   Absent    00   DEAD          0    00000000
4    5   Absent    00   DEAD          0    00000000
4    6   Absent    00   DEAD          0    00000000
4    8   Absent    00   DEAD          0    00000000
4    9   Absent    00   DEAD          0    00000000
4    a   Absent    00   DEAD          0    00000000
4    b   Absent    00   DEAD          0    00000000
4    c   Absent    00   DEAD          0    00000000
4    d   Absent    00   DEAD          0    00000000
4    e   Absent    00   DEAD          0    00000000
4    f   Absent    00   DEAD          0    00000000
```

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

    Hardware

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

    Physical

Default rebuild rate                 50
Controller read ahead             Enabled
Super read ahead                 Disabled
True Verification of Data        Disabled
Stripe Size(K bytes)                 64
Installation Abort               Disabled
Reassign Restricted to 1 blk      Enabled
Write Through Verify              Enabled
RAID 5 Algorithm               Right Asym

    Disk Side Parameters

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

    Host Side

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

    Serial port 0
```

```
    Active - Active

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

    Fibre

  PCI Latency Control                Short
  Frame Control                       Long
  Host Algorithm Control          As Avail
  Port 0                          Disabled
  Port 0 ID                              0
  Port 1                          Disabled
  Port 1 ID                              0

   ************************************************************************
   *        MYLEX Disk Array Controller - Configuration Utility           *
   *                            Version 7.09                              *
   ************************************************************************

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

   5 Channel - 16 Target  DAC960SX  Firmware version 3.24-00

   Daughter Card


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

   Number of Packs = 1

   Pack A : [2:0]   [2:1]   [2:2]   [2:3]   [2:8]   [2:9]   [2:a]   [2:b]
```

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

Number of System Drives = 1
LUN Affinity = Disabled

# Phy. Size  Raid  Eff. Size  Cache       Packs  Status
= =========  ====  =========  ==========  =====  ======
0  69424 MB    0    69424 MB  Write Back  A...   Onln

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

All Affinity (No LUN mapping)

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

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

```
1    9   Absent    00   DEAD          0    00000000
1    a   Absent    00   DEAD          0    00000000
1    b   Absent    00   DEAD          0    00000000
1    c   Absent    00   DEAD          0    00000000
1    d   Absent    00   DEAD          0    00000000
1    e   Absent    00   DEAD          0    00000000
1    f   Absent    00   DEAD          0    00000000
2    0   Present   f1   ONLINE     8678    010f3000
2    1   Present   f1   ONLINE     8678    010f3000
2    2   Present   f1   ONLINE     8678    010f3000
2    3   Present   f1   ONLINE     8678    010f3000
2    4   Absent    00   DEAD          0    00000000
2    5   Absent    00   DEAD          0    00000000
2    6   Absent    00   DEAD          0    00000000
2    8   Present   f1   ONLINE     8678    010f3000
2    9   Present   f1   ONLINE     8678    010f3000
2    a   Present   f1   ONLINE     8678    010f3000
2    b   Present   f1   ONLINE     8678    010f3000
2    c   Absent    00   DEAD          0    00000000
2    d   Absent    00   DEAD          0    00000000
2    e   Absent    00   DEAD          0    00000000
2    f   Absent    00   DEAD          0    00000000
3    0   Absent    00   DEAD          0    00000000
3    1   Absent    00   DEAD          0    00000000
3    2   Absent    00   DEAD          0    00000000
3    3   Absent    00   DEAD          0    00000000
3    4   Absent    00   DEAD          0    00000000
3    5   Absent    00   DEAD          0    00000000
3    6   Absent    00   DEAD          0    00000000
3    8   Absent    00   DEAD          0    00000000
3    9   Absent    00   DEAD          0    00000000
3    a   Absent    00   DEAD          0    00000000
3    b   Absent    00   DEAD          0    00000000
3    c   Absent    00   DEAD          0    00000000
3    d   Absent    00   DEAD          0    00000000
3    e   Absent    00   DEAD          0    00000000
3    f   Absent    00   DEAD          0    00000000
4    0   Absent    00   DEAD          0    00000000
4    1   Absent    00   DEAD          0    00000000
4    2   Absent    00   DEAD          0    00000000
4    3   Absent    00   DEAD          0    00000000
4    4   Absent    00   DEAD          0    00000000
4    5   Absent    00   DEAD          0    00000000
4    6   Absent    00   DEAD          0    00000000
4    8   Absent    00   DEAD          0    00000000
4    9   Absent    00   DEAD          0    00000000
4    a   Absent    00   DEAD          0    00000000
4    b   Absent    00   DEAD          0    00000000
4    c   Absent    00   DEAD          0    00000000
4    d   Absent    00   DEAD          0    00000000
4    e   Absent    00   DEAD          0    00000000
4    f   Absent    00   DEAD          0    00000000

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

    Hardware

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

    Physical

Default rebuild rate                 50
Controller read ahead             Enabled
Super read ahead                 Disabled
True Verification of Data        Disabled
Stripe Size(K bytes)                 64
Installation Abort               Disabled
Reassign Restricted to 1 blk      Enabled
Write Through Verify              Enabled
RAID 5 Algorithm               Right Asym

    Disk Side Parameters

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

```
    Host Side

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

    Serial port 0



    Active - Active

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

    Fibre

PCI Latency Control                Short
Frame Control                       Long
Host Algorithm Control          As Avail
Port 0                          Disabled
Port 0 ID                              0
Port 1                          Disabled
Port 1 ID                              0

 ************************************************************************
 *        MYLEX Disk Array Controller - Configuration Utility           *
 *                            Version 7.09                              *
 ************************************************************************

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

 5 Channel - 16 Target  DAC960SX  Firmware version 3.24-00

 Daughter Card
```

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

Number of Packs = 1

Pack A : [2:0]  [2:1]  [2:2]  [2:3]  [2:8]  [2:9]  [2:a]  [2:b]

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

Number of System Drives = 1
LUN Affinity = Disabled

# Phy. Size  Raid  Eff. Size  Cache       Packs  Status
= =========  ====  =========  ==========  =====  ======
0  69424 MB    0    69424 MB  Write Back  A...   Onln

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

All Affinity (No LUN mapping)

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

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

```
1    0   Absent    00   DEAD          0    00000000
1    1   Absent    00   DEAD          0    00000000
1    2   Absent    00   DEAD          0    00000000
1    3   Absent    00   DEAD          0    00000000
1    4   Absent    00   DEAD          0    00000000
1    5   Absent    00   DEAD          0    00000000
1    6   Absent    00   DEAD          0    00000000
1    8   Absent    00   DEAD          0    00000000
1    9   Absent    00   DEAD          0    00000000
1    a   Absent    00   DEAD          0    00000000
1    b   Absent    00   DEAD          0    00000000
1    c   Absent    00   DEAD          0    00000000
1    d   Absent    00   DEAD          0    00000000
1    e   Absent    00   DEAD          0    00000000
1    f   Absent    00   DEAD          0    00000000
2    0   Present   f1   ONLINE     8678    010f3000
2    1   Present   f1   ONLINE     8678    010f3000
2    2   Present   f1   ONLINE     8678    010f3000
2    3   Present   f1   ONLINE     8678    010f3000
2    4   Absent    00   DEAD          0    00000000
2    5   Absent    00   DEAD          0    00000000
2    6   Absent    00   DEAD          0    00000000
2    8   Present   f1   ONLINE     8678    010f3000
2    9   Present   f1   ONLINE     8678    010f3000
2    a   Present   f1   ONLINE     8678    010f3000
2    b   Present   f1   ONLINE     8678    010f3000
2    c   Absent    00   DEAD          0    00000000
2    d   Absent    00   DEAD          0    00000000
2    e   Absent    00   DEAD          0    00000000
2    f   Absent    00   DEAD          0    00000000
3    0   Absent    00   DEAD          0    00000000
3    1   Absent    00   DEAD          0    00000000
3    2   Absent    00   DEAD          0    00000000
3    3   Absent    00   DEAD          0    00000000
3    4   Absent    00   DEAD          0    00000000
3    5   Absent    00   DEAD          0    00000000
3    6   Absent    00   DEAD          0    00000000
3    8   Absent    00   DEAD          0    00000000
3    9   Absent    00   DEAD          0    00000000
3    a   Absent    00   DEAD          0    00000000
3    b   Absent    00   DEAD          0    00000000
3    c   Absent    00   DEAD          0    00000000
3    d   Absent    00   DEAD          0    00000000
3    e   Absent    00   DEAD          0    00000000
3    f   Absent    00   DEAD          0    00000000
4    0   Absent    00   DEAD          0    00000000
4    1   Absent    00   DEAD          0    00000000
4    2   Absent    00   DEAD          0    00000000
4    3   Absent    00   DEAD          0    00000000
4    4   Absent    00   DEAD          0    00000000
4    5   Absent    00   DEAD          0    00000000
4    6   Absent    00   DEAD          0    00000000
4    8   Absent    00   DEAD          0    00000000
4    9   Absent    00   DEAD          0    00000000
4    a   Absent    00   DEAD          0    00000000
4    b   Absent    00   DEAD          0    00000000
4    c   Absent    00   DEAD          0    00000000
4    d   Absent    00   DEAD          0    00000000
4    e   Absent    00   DEAD          0    00000000
4    f   Absent    00   DEAD          0    00000000

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

    Hardware

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

    Physical

Default rebuild rate                 50
Controller read ahead             Enabled
Super read ahead                 Disabled
True Verification of Data        Disabled
Stripe Size(K bytes)                 64
Installation Abort               Disabled
Reassign Restricted to 1 blk      Enabled
Write Through Verify              Enabled
RAID 5 Algorithm               Right Asym

    Disk Side Parameters

Per Channel:            0     1     2     3     4
 Data Transfer Rate - 10MHz 10MHz 10MHz 10MHz 10MHz
 Command tagging    - Enabl Enabl Enabl Enabl Enabl
 Data bus width     - 16bit 16bit 16bit 16bit 16bit
```

```
Per Device:
 Elevator   -  Disab
 Coalescing -  Disab
 Que Limit  -    0
Global:
 Max IOPs       -   0
 Spin Up Option - Automatic  /  2 /  6 /  0

    Host Side

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

    Serial port 0



    Active - Active

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

    Fibre

PCI Latency Control                Short
Frame Control                       Long
Host Algorithm Control          As Avail
Port 0                          Disabled
Port 0 ID                             0
Port 1                          Disabled
Port 1 ID                             0
```

C.8 Client System Configuration Parameters

```
Microsoft Diagnostics Report For \\UWLC1
------------------------------------------------------------------------

OS Version Report
------------------------------------------------------------------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiprocessor Free
Registered Owner: Hewlett Packard, Network Server Division
Product Number: 50370-111-1111111-91655
------------------------------------------------------------------------

System Report
------------------------------------------------------------------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 07/18/98
BIOS Version: <unavailable>

Processor list:
   0:  x86 Family 6 Model 5 Stepping 1 GenuineIntel ~399 Mhz
   1:  x86 Family 6 Model 5 Stepping 1 GenuineIntel ~399 Mhz
------------------------------------------------------------------------

Video Display Report
------------------------------------------------------------------------
BIOS Date: 05/21/97
BIOS Version: CL-GD5446 PCI VGA BIOS Version 1.33

Adapter:
   Setting: 800 x 600 x 16
            Hardware Default Refresh
   Type: vga compatible display adapter
   String: <unavailable>
   Memory:
   Chip Type: <unavailable>
   DAC Type: <unavailable>
Driver:
   Vendor: Microsoft Corporation
   File(s): vga.sys, vga.dll
   Version: 4.00, 4.0.0


Drives Report
------------------------------------------------------------------------
C:\  (Local - NTFS) TPCC_CLIENT Total: 0KB, Free: 0KB
   Serial Number: 363B - 18E9
```

```
   Bytes per cluster: 512
   Sectors per cluster: 1
   Filename length: 255
E:  (Remote - NTFS) \\nrddata\g$  Total: 44,431,036KB, Free: 18,034,412KB
   Serial Number: C0EF - 452A
   Bytes per cluster: 512
   Sectors per cluster: 8
   Filename length: 255

Memory Report
-------------------------------------------------------------------------
Handles: 20,277
Threads: 149
Processes: 19

Physical Memory (K)
   Total: 523,696
   Available: 441,416
   File Cache: 15,632

Kernel Memory (K)
   Total: 24,716
   Paged: 9,112
   Nonpaged: 15,604

Commit Charge (K)
   Total: 54,548
   Limit: 904,556
   Peak: 54,648

Pagefile Space (K)
   Total: 409,600
   Total in use: 192
   Peak: 192

   C:\pagefile.sys
      Total: 409,600
      Total in use: 192
      Peak: 192



Services Report
-------------------------------------------------------------------------
Alerter                                          Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
Ataman TCP Remote Logon Services                 Running   (Automatic)
   C:\atrls2\ATRLS.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Computer Browser                                 Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
      LmHosts
ClipBook Server                                  Stopped   (Manual)
   C:\WINNT\system32\clipsrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      NetDDE
DHCP Client (TDI)                                Stopped   (Disabled)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      Tcpip
      Afd
      NetBT
EventLog (Event log)                             Running   (Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Gopher Publishing Service                        Running   (Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
```

```
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
Server                                  Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
Workstation (NetworkProvider)           Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      TDI
License Logging Service                 Running   (Automatic)
   C:\WINNT\System32\llssrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TCP/IP NetBIOS Helper                   Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Group Dependencies:
      NetworkProvider
Messenger                               Running   (Automatic)
   C:\WINNT\System32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      NetBios
Monitor Service                         Stopped   (Manual)
   C:\WINNT\system32\datalog.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
FTP Publishing Service                  Running   (Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP
MSSQLServer                             Stopped   (Disabled)
   c:\MSSQL\BINN\SQLSERVR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
Network DDE (NetDDEGroup)               Stopped   (Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      NetDDEDSDM
Network DDE DSDM                        Stopped   (Manual)
   C:\WINNT\system32\netdde.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Net Logon (RemoteValidation)            Stopped   (Manual)
   C:\WINNT\System32\lsass.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
   Service Dependencies:
      LanmanWorkstation
      LmHosts
Network Monitor Agent                   Stopped   (Manual)
   C:\WINNT\System32\nmagent.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      bh
NT LM Security Support Provider         Running   (Manual)
   C:\WINNT\System32\SERVICES.EXE
   Service Account Name: LocalSystem
```

```
   Error Severity: Normal
   Service Flags: Shared Process
Plug and Play (PlugPlay)                    Running   (Automatic)
   C:\WINNT\system32\services.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Shared Process
Directory Replicator                        Stopped   (Manual)
   C:\WINNT\System32\lmrepl.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      LanmanServer
Remote Command Service                      Running   (Automatic)
   c:\benchcrf\rsys.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
Remote Procedure Call (RPC) Locator         Stopped   (Manual)
   C:\WINNT\System32\LOCATOR.EXE
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
   Service Dependencies:
      LanmanWorkstation
      Rdr
Remote Procedure Call (RPC) Service         Running   (Automatic)
   C:\WINNT\system32\RpcSs.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Schedule                                    Stopped   (Manual)
   C:\WINNT\System32\AtSvc.Exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
Spooler (SpoolerGroup)                      Running   (Automatic)
   C:\WINNT\system32\spoolss.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
SQLExecutive                                Stopped   (Manual)
   C:\MSSQL\BINN\SQLEXEC.EXE
   Service Account Name: .\Administrator
   Error Severity: Normal
   Service Flags: Own Process
Telephony Service                           Stopped   (Manual)
   C:\WINNT\system32\tapisrv.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TUXEDO IPC HELPER                           Running   (Automatic)
   C:\TUXEDO\bin\tuxipc.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
TListen (Port: 3050)                        Running   (Automatic)
   C:\TUXEDO\bin\slisten.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process
UPS                                         Stopped   (Automatic)
   C:\WINNT\System32\ups.exe
   Service Account Name: LocalSystem
   Error Severity: Normal
   Service Flags: Own Process, Interactive
World Wide Web Publishing Service           Running   (Automatic)
   C:\WINNT\System32\inetsrv\inetinfo.exe
   Service Account Name: LocalSystem
   Error Severity: Ignore
   Service Flags: Shared Process
   Service Dependencies:
      RPCSS
      NTLMSSP


Drivers Report
--------------------------------------------------------------------------------
Abiosdsk (Primary disk)                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
AFD Networking Support Environment (TDI)    Running   (Automatic)
   C:\WINNT\System32\drivers\afd.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
```

```
Aha154x (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Aha174x (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
aic78xx (SCSI miniport)                          Running   (Boot)
   C:\WINNT\System32\DRIVERS\aic78xx.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Always (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ami0nt (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
amsint (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Arrow (SCSI miniport)                            Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
atapi (SCSI miniport)                            Running   (Boot)
   C:\WINNT\System32\DRIVERS\atapi.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Atdisk (Primary disk)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ati (Video)                                      Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Beep (Base)                                      Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Network Monitor Tools and Agent Drivers          Stopped   (Manual)
   C:\WINNT\System32\drivers\bhnt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      NDIS
BusLogic (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Busmouse (Pointer Port)                          Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdaudio (Filter)                                 Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cdfs (File system)                               Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      SCSI CDROM Class
Cdrom (SCSI CDROM Class)                         Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Changer (Filter)                                 Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
cirrus (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Cpqarray (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
cpqfws2e (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dac960nt (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dce376nt (SCSI miniport)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Delldsa (SCSI miniport)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Dell_DGX (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Disk (SCSI Class)                                Running   (Boot)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
```

```
      SCSI miniport
Diskperf (Filter)                                  Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
DptScsi (SCSI miniport)                            Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
dtc329x (SCSI miniport)                            Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
et4000 (Video)                                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Fastfat (Boot file system)                         Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Fd16_700 (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd7000ex (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Fd8xx (SCSI miniport)                              Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
flashpnt (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Floppy (Primary disk)                              Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ftdisk (Filter)                                    Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
HP 10/100TX PCI Ethernet Adapter Driver (NDIS) Running    (Automatic)
   C:\WINNT\System32\drivers\hptxnt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
i8042 Keyboard and PS/2 Mouse Port Driver (Keyboard Port) Running   (System)
   System32\DRIVERS\i8042prt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Inport (Pointer Port)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg300 (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jazzg364 (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Jzvxl484 (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Keyboard Class Driver (Keyboard Class)             Running   (System)
   System32\DRIVERS\kbdclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
KSecDD (Base)                                      Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mga (Video)                                        Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mga64 (Video)                                      Stopped   (System)
   System32\DRIVERS\mga64.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
mga_mil (Video)                                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mitsumi (SCSI miniport)                            Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mkecr5xx (SCSI miniport)                           Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Modem (Extended base)                              Stopped   (Manual)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Mouse Class Driver (Pointer Class)                 Running   (System)
   System32\DRIVERS\mouclass.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Msfs (File system)                                 Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                                      Running   (Manual)
```

```
   C:\WINNT\System32\drivers\mup.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ncr53c9x (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
ncr77c22 (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ncrc700 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ncrc710 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Microsoft NDIS System Driver (NDIS)         Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
NetBIOS Interface (NetBIOSGroup)            Running   (Manual)
   C:\WINNT\System32\drivers\netbios.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
   Group Dependencies:
      TDI
WINS Client(TCP/IP) (PNP_TDI)               Running   (Automatic)
   C:\WINNT\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                                   Stopped   (Manual)
   C:\WINNT\system32\drivers\netdtect.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Npfs (File system)                          Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Ntfs (File system)                          Running   (Disabled)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Null (Base)                                 Running   (System)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Oliscsi (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Parallel (Extended base)                    Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
Parport (Parallel arbitrator)               Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
ParVdm (Extended base)                      Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Parport
   Group Dependencies:
      Parallel arbitrator
PCIDump (PCI Configuration)                 Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Pcmcia (System Bus Extender)                Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
PnP ISA Enabler Driver (Base)               Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
psidisp (Video)                             Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Ql10wnt (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
qv (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Rdr (Network)                               Running   (Manual)
   C:\WINNT\System32\drivers\rdr.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
s3 (Video)                                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
```

```
Scsiprnt (Extended base)                    Stopped   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Scsiscan (SCSI Class)                       Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Serial (Extended base)                      Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sermouse (Pointer Port)                     Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sfloppy (Primary disk)                      Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Simbad (Filter)                             Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
slcd32 (SCSI miniport)                      Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Sparrow (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Spock (SCSI miniport)                       Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Srv (Network)                               Running   (Manual)
   C:\WINNT\System32\drivers\srv.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
symc810 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T128 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T13B (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
TCP/IP Service (PNP_TDI)                    Running   (Automatic)
   C:\WINNT\System32\drivers\tcpip.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
tga (Video)                                 Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
tmv1 (SCSI miniport)                        Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra124 (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra14f (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Ultra24f (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
v7vram (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
vga (Video)                                 Stopped   (Disabled)
   System32\DRIVERS\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaSave (Video Save)                        Running   (System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                       Stopped   (System)
   C:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Wd33c93 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
wd90c24a (Video)                            Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
wdvga (Video)                               Stopped   (Disabled)
   Error Severity: Ignore
```

```
   Service Flags: Kernel Driver, Shared Process
weitekp9 (Video)                              Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Xga (Video)                                   Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
```

IRQ and Port Report

Devices	Vector	Level	Affinity
MPS 1.4 - APIC platform	8	8	0x00000003
MPS 1.4 - APIC platform	0	0	0x00000003
MPS 1.4 - APIC platform	1	1	0x00000003
MPS 1.4 - APIC platform	2	2	0x00000003
MPS 1.4 - APIC platform	3	3	0x00000003
MPS 1.4 - APIC platform	4	4	0x00000003
MPS 1.4 - APIC platform	5	5	0x00000003
MPS 1.4 - APIC platform	6	6	0x00000003
MPS 1.4 - APIC platform	7	7	0x00000003
MPS 1.4 - APIC platform	8	8	0x00000003
MPS 1.4 - APIC platform	9	9	0x00000003
MPS 1.4 - APIC platform	10	10	0x00000003
MPS 1.4 - APIC platform	11	11	0x00000003
MPS 1.4 - APIC platform	12	12	0x00000003
MPS 1.4 - APIC platform	13	13	0x00000003
MPS 1.4 - APIC platform	14	14	0x00000003
MPS 1.4 - APIC platform	15	15	0x00000003
MPS 1.4 - APIC platform	16	16	0x00000003
MPS 1.4 - APIC platform	17	17	0x00000003
MPS 1.4 - APIC platform	18	18	0x00000003
MPS 1.4 - APIC platform	19	19	0x00000003
MPS 1.4 - APIC platform	20	20	0x00000003
MPS 1.4 - APIC platform	21	21	0x00000003
MPS 1.4 - APIC platform	22	22	0x00000003
MPS 1.4 - APIC platform	23	23	0x00000003
MPS 1.4 - APIC platform	24	24	0x00000003
MPS 1.4 - APIC platform	25	25	0x00000003
MPS 1.4 - APIC platform	26	26	0x00000003
MPS 1.4 - APIC platform	27	27	0x00000003
MPS 1.4 - APIC platform	28	28	0x00000003
MPS 1.4 - APIC platform	29	29	0x00000003
MPS 1.4 - APIC platform	30	30	0x00000003
MPS 1.4 - APIC platform	31	31	0x00000003
MPS 1.4 - APIC platform	32	32	0x00000003
MPS 1.4 - APIC platform	33	33	0x00000003
MPS 1.4 - APIC platform	34	34	0x00000003
MPS 1.4 - APIC platform	35	35	0x00000003
MPS 1.4 - APIC platform	36	36	0x00000003
MPS 1.4 - APIC platform	37	37	0x00000003
MPS 1.4 - APIC platform	38	38	0x00000003
MPS 1.4 - APIC platform	39	39	0x00000003
MPS 1.4 - APIC platform	40	40	0x00000003
MPS 1.4 - APIC platform	41	41	0x00000003
MPS 1.4 - APIC platform	42	42	0x00000003
MPS 1.4 - APIC platform	43	43	0x00000003
MPS 1.4 - APIC platform	44	44	0x00000003
MPS 1.4 - APIC platform	45	45	0x00000003
MPS 1.4 - APIC platform	46	46	0x00000003
MPS 1.4 - APIC platform	47	47	0x00000003
MPS 1.4 - APIC platform	61	61	0x00000003
MPS 1.4 - APIC platform	65	65	0x00000003
MPS 1.4 - APIC platform	80	80	0x00000003
MPS 1.4 - APIC platform	193	193	0x00000003
MPS 1.4 - APIC platform	225	225	0x00000003
MPS 1.4 - APIC platform	253	253	0x00000003
MPS 1.4 - APIC platform	254	254	0x00000003
MPS 1.4 - APIC platform	255	255	0x00000003
i8042prt	1	1	0xffffffff
i8042prt	12	12	0xffffffff
Serial	4	4	0x00000000
Serial	3	3	0x00000000
Floppy	6	6	0x00000000
HPTX	32	32	0x00000000
aic78xx	40	40	0x00000000
atapi	0	14	0x00000000

Devices	Physical Address	Length
MPS 1.4 - APIC platform	0x00000000	0x0000000010
MPS 1.4 - APIC platform	0x00000020	0x0000000002
MPS 1.4 - APIC platform	0x00000040	0x0000000004
MPS 1.4 - APIC platform	0x00000048	0x0000000004
MPS 1.4 - APIC platform	0x00000061	0x0000000001
MPS 1.4 - APIC platform	0x00000070	0x0000000002

```
MPS 1.4 - APIC platform         0x00000080  0x0000000010
MPS 1.4 - APIC platform         0x00000092  0x0000000001
MPS 1.4 - APIC platform         0x000000a0  0x0000000002
MPS 1.4 - APIC platform         0x000000c0  0x0000000010
MPS 1.4 - APIC platform         0x000000f0  0x0000000010
i8042prt                        0x00000060  0x0000000001
i8042prt                        0x00000064  0x0000000001
Parport                         0x00000378  0x0000000003
Serial                          0x000003f8  0x0000000007
Serial                          0x000002f8  0x0000000007
Floppy                          0x000003f0  0x0000000006
Floppy                          0x000003f7  0x0000000001
HPTX                            0x0000fcc0  0x0000000014
aic78xx                         0x0000f800  0x0000000100
atapi                           0x000001f0  0x0000000008
atapi                           0x000003f6  0x0000000001
VgaSave                         0x000003b0  0x000000000c
VgaSave                         0x000003c0  0x0000000020
VgaSave                         0x000001ce  0x0000000002



DMA and Memory Report
-------------------------------------------------------------------------
Devices                         Channel    Port
-------------------------------------------------------------------------
Floppy                             2     0
-------------------------------------------------------------------------
Devices                         Physical Address  Length
-------------------------------------------------------------------------
MPS 1.4 - APIC platform         0xfec00000  0x00000400
MPS 1.4 - APIC platform         0xfee00000  0x00000400
HPTX                            0xfecfd000  0x00000014
aic78xx                         0xfecff000  0x00001000
VgaSave                         0x000a0000  0x00020000



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



System Environment Variables
   ComSpec=C:\WINNT\system32\cmd.exe
   HOME=C:/
   NTRESKIT=C:\NTRESKIT
   NUMBER_OF_PROCESSORS=2
   OS=Windows_NT
   Os2LibPath=C:\WINNT\system32\os2\dll;

Path=C:\mksnt;C:\WINNT\system32;C:\WINNT;;C:\MSSQL\BINN;C:\TUXEDO\bin;C:\NTRESK
IT
   PROCESSOR_ARCHITECTURE=x86
   PROCESSOR_IDENTIFIER=x86 Family 6 Model 5 Stepping 1, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=0501
   ROOTDIR=C:/
   SHELL=C:/mksnt/sh.exe
   TMPDIR=C:/TEMP
   windir=C:\WINNT
   TUXDIR=C:\TUXEDO
   APPDIR=C:\inetpub\wwwroot
   TMCONTEXTS=1
   TUXCONFIG=C:\inetpub\wwwroot\tuxconfig



Environment Variables for Current User

Path=C:\mksnt;C:\WINNT\system32;C:\WINNT;C:\MSSQL\BINN;C:\TUXEDO\bin;C:\NTRESKI
T
   TEMP=C:\TEMP
   TMP=C:\TEMP



Network Report
-------------------------------------------------------------------------
Your Access Level: Admin & Local
Workgroup or Domain: WORKGROUP
Network Version: 4.0
LanRoot: WORKGROUP
Logged On Users: 2
Current User (1): Administrator
  Logon Domain: UWLC1
  Logon Server: UWLC1
Current User (2): Administrator
```

```
  Logon Domain: UWLC1
  Logon Server: UWLC1

Transport: NetBT_HPTX1, 08-00-09-DC-0A-A9, VC's: 1, Wan: Wan

Character Wait: 3,600
Collection Time: 250
Maximum Collection Count: 16
Keep Connection: 600
Maximum Commands: 5
Session Time Out: 45
Character Buffer Size: 512
Maximum Threads: 17
Lock Quota: 6,144
Lock Increment: 10
Maximum Locks: 500
Pipe Increment: 10
Maximum Pipes: 500
Cache Time Out: 40
Dormant File Limit: 45
Read Ahead Throughput: 4,294,967,295
Mailslot Buffers: 3
Server Announce Buffers: 20
Illegal Datagrams: 5
Datagram Reset Frequency: 60
Log Election Packets: False
Use Opportunistic Locking: True
Use Unlock Behind: True
Use Close Behind: True
Buffer Pipes: True
Use Lock, Read, Unlock: True
Use NT Caching: True
Use Raw Read: True
Use Raw Write: True
Use Write Raw Data: True
Use Encryption: True
Buffer Deny Write Files: True
Buffer Read Only Files: True
Force Core Creation: True
512 Byte Max Transfer: False
Bytes Received: 10,990
SMB's Received: 75
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 0
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Bytes Transmitted: 7,722
SMB's Transmitted: 75
Paged Read Bytes Requested: 0
Non Paged Read Bytes Requested: 364
Cache Read Bytes Requested: 0
Network Read Bytes Requested: 0
Initally Failed Operations: 0
Failed Completion Operations: 0
Read Operations: 0
Random Read Operations: 0
Read SMB's: 0
Large Read SMB's: 0
Small Read SMB's: 0
Write Operations: 7
Random Write Operations: 0
Write SMB's: 0
Large Write SMB's: 0
Small Write SMB's: 0
Raw Reads Denied: 0
Raw Writes Denied: 0
Network Errors: 0
Sessions: 3
Failed Sessions: 0
Reconnects: 0
Core Connects: 0
LM 2.0 Connects: 0
LM 2.x Connects: 0
Windows NT Connects: 2
Server Disconnects: 0
Hung Sessions: 0
Use Count: 2
Failed Use Count: 0
Current Commands: 0
Server File Opens: 362
Server Device Opens: 0
Server Jobs Queued: 0
Server Session Opens: 1
Server Sessions Timed Out: 0
Server Sessions Errored Out: 0
Server Password Errors: 1
Server Permission Errors: 0
Server System Errors: 0
```

```
Server Bytes Sent: 877,001
Server Bytes Received: 121,308
Server Average Response Time: 0
Server Request Buffers Needed: 0
Server Big Buffers Needed: 0
```

C.9 NT Registry

Bea

```
Key Name:          SOFTWARE\BEA Systems
Class Name:        <NO CLASS>
Last Write Time:   12/9/97 - 3:24 PM

Key Name:          SOFTWARE\BEA Systems\Tuxedo
Class Name:        <NO CLASS>
Last Write Time:   12/9/97 - 3:24 PM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM
Value 0
  Name:            Company_Name
  Type:            REG_SZ
  Data:            Network Server Division

Value 1
  Name:            Install_Date
  Type:            REG_SZ
  Data:            1-2-1997

Value 2
  Name:            License_Token
  Type:            REG_DWORD
  Data:            0

Value 3
  Name:            Major_Version
  Type:            REG_DWORD
  Data:            0x6

Value 4
  Name:            Minor_Version
  Type:            REG_DWORD
  Data:            0x4

Value 5
  Name:            Serial_Number
  Type:            REG_DWORD
  Data:            0

Value 6
  Name:            User_Name
  Type:            REG_SZ
  Data:            Hewlett Packard

Value 7
  Name:            Volume_Number
  Type:            REG_DWORD
  Data:            0x1


Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Developer
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Developer\Libraries
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Developer\Libraries\All
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\All\libfml.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\All\libfml32.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM
```

```
Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\All\libgp.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Developer\Libraries\Client
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Client\libbuft.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Client\libtux.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Client\libtux2.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Developer\Libraries\Server
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Server\libbuft.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Server\libtux.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Server\libtux2.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Workstation
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Workstation\libbuft.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Workstation\libnwi.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Workstation\libnws.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA
Systems\Tuxedo\6.4\Developer\Libraries\Workstation\libwsc.lib
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Environment
Class Name:        <NO CLASS>
Last Write Time:   6/25/97 - 6:47 PM
Value 0
  Name:            NLSPATH
  Type:            REG_SZ
  Data:            C:\TUXEDO\locale\C

Value 1
  Name:            TUXDIR
  Type:            REG_SZ
  Data:            C:\TUXEDO

Value 2
  Name:            TUXIPC_MSG_BYTES
  Type:            REG_DWORD
  Data:            0x10000

Value 3
```

```
  Name:            TUXIPC_MSG_HDRS
  Type:            REG_DWORD
  Data:            0x1fc0

Value 4
  Name:            TUXIPC_MSG_QUEUE_BYTES
  Type:            REG_DWORD
  Data:            0x10000

Value 5
  Name:            TUXIPC_MSG_QUEUES
  Type:            REG_DWORD
  Data:            0x400

Value 6
  Name:            TUXIPC_MSG_SEG_BYTES
  Type:            REG_DWORD
  Data:            0x40

Value 7
  Name:            TUXIPC_MSG_SEGS
  Type:            REG_DWORD
  Data:            0x7fff

Value 8
  Name:            TUXIPC_PROC
  Type:            REG_DWORD
  Data:            0x400

Value 9
  Name:            TUXIPC_SEM
  Type:            REG_DWORD
  Data:            0x800

Value 10
  Name:            TUXIPC_SEM_IDS
  Type:            REG_DWORD
  Data:            0x800

Value 11
  Name:            TUXIPC_SEM_UNDO
  Type:            REG_DWORD
  Data:            0x800

Value 12
  Name:            TUXIPC_SHM_PROCS
  Type:            REG_DWORD
  Data:            0x800

Value 13
  Name:            TUXIPC_SHM_SEGS
  Type:            REG_DWORD
  Data:            0x32

Value 14
  Name:            ULOGDIR
  Type:            REG_SZ
  Data:            C:\INETPUB\WWWROOT

Value 15
  Name:            ULOGOUT
  Type:            REG_DWORD
  Data:            0x2

Value 16
  Name:            ULOGPFX
  Type:            REG_SZ
  Data:            C:\ULOG


Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Environment\Services
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\Environment\Services\3050
Class Name:        <NO CLASS>
Last Write Time:   1/2/97 - 6:17 AM

Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\IPCResources
Class Name:        <NO CLASS>
Last Write Time:   6/25/97 - 6:47 PM
Value 0
  Name:            CurrentResource
  Type:            REG_SZ
  Data:            tpcc


Key Name:          SOFTWARE\BEA Systems\Tuxedo\6.4\IPCResources\tpcc
```

```
Class Name:         <NO CLASS>
Last Write Time:    6/25/97 - 6:47 PM
Value 0
  Name:             TUXIPC_MSG_BYTES
  Type:             REG_DWORD
  Data:             0x10000

Value 1
  Name:             TUXIPC_MSG_HDRS
  Type:             REG_DWORD
  Data:             0x1fc0

Value 2
  Name:             TUXIPC_MSG_QUEUE_BYTES
  Type:             REG_DWORD
  Data:             0x10000

Value 3
  Name:             TUXIPC_MSG_QUEUES
  Type:             REG_DWORD
  Data:             0x400

Value 4
  Name:             TUXIPC_MSG_SEG_BYTES
  Type:             REG_DWORD
  Data:             0x40

Value 5
  Name:             TUXIPC_MSG_SEGS
  Type:             REG_DWORD
  Data:             0x7fff

Value 6
  Name:             TUXIPC_PROC
  Type:             REG_DWORD
  Data:             0x400

Value 7
  Name:             TUXIPC_SEM
  Type:             REG_DWORD
  Data:             0x800

Value 8
  Name:             TUXIPC_SEM_IDS
  Type:             REG_DWORD
  Data:             0x800

Value 9
  Name:             TUXIPC_SEM_UNDO
  Type:             REG_DWORD
  Data:             0x800

Value 10
  Name:             TUXIPC_SHM_PROCS
  Type:             REG_DWORD
  Data:             0x800

Value 11
  Name:             TUXIPC_SHM_SEGS
  Type:             REG_DWORD
  Data:             0x32


Key Name:           SOFTWARE\BEA Systems\Tuxedo\6.4\SECURITY
Class Name:         <NO CLASS>
Last Write Time:    1/2/97 - 6:17 AM

Key Name:           SOFTWARE\BEA Systems\Tuxedo\Release 6.2
Class Name:         <NO CLASS>
Last Write Time:    12/9/97 - 3:24 PM
Value 0
  Name:             INSTALLDATE
  Type:             REG_SZ
  Data:             12/09/97

Value 1
  Name:             MAJORVERSION
  Type:             REG_DWORD
  Data:             0x6

Value 2
  Name:             MINORVERSION
  Type:             REG_DWORD
  Data:             0x2

Value 3
  Name:             SERIALNUMBER
  Type:             REG_SZ
```

```
  Data:            9701100012

Value 4
  Name:            VOLUMEVERSION
  Type:            REG_DWORD
  Data:            0x1


Key Name:          SOFTWARE\BEA Systems\Tuxedo\Release 6.2\Environment
Class Name:        <NO CLASS>
Last Write Time:   12/9/97 - 3:24 PM
Value 0
  Name:            NLSPATH
  Type:            REG_SZ
  Data:            C:\TUXEDO\locale\C

Value 1
  Name:            TUXDIR
  Type:            REG_SZ
  Data:            C:\TUXEDO

Value 2
  Name:            TUXIPC_MSG_BYTES
  Type:            REG_DWORD
  Data:            0x10000

Value 3
  Name:            TUXIPC_MSG_HDRS
  Type:            REG_DWORD
  Data:            0x1fc0

Value 4
  Name:            TUXIPC_MSG_QUEUE_BYTES
  Type:            REG_DWORD
  Data:            0x10000

Value 5
  Name:            TUXIPC_MSG_QUEUES
  Type:            REG_DWORD
  Data:            0x100

Value 6
  Name:            TUXIPC_MSG_SEG_BYTES
  Type:            REG_DWORD
  Data:            0x40

Value 7
  Name:            TUXIPC_MSG_SEGS
  Type:            REG_DWORD
  Data:            0x7fff

Value 8
  Name:            TUXIPC_PROC
  Type:            REG_DWORD
  Data:            0x100

Value 9
  Name:            TUXIPC_SEM
  Type:            REG_DWORD
  Data:            0x400

Value 10
  Name:            TUXIPC_SEM_IDS
  Type:            REG_DWORD
  Data:            0x400

Value 11
  Name:            TUXIPC_SEM_UNDO
  Type:            REG_DWORD
  Data:            0x400

Value 12
  Name:            TUXIPC_SHM_PROCS
  Type:            REG_DWORD
  Data:            0x1f4

Value 13
  Name:            TUXIPC_SHM_SEGS
  Type:            REG_DWORD
  Data:            0x32

Value 14
  Name:            ULOGPFX
  Type:            REG_SZ
  Data:            C:\ULOG
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session Manager
Class Name:        <NO CLASS>
Last Write Time:   10/26/98 - 3:09 PM
Value 0
  Name:            BootExecute
  Type:            REG_MULTI_SZ
  Data:            autocheck autochk *

Value 1
  Name:            CriticalSectionTimeout
  Type:            REG_DWORD
  Data:            0x278d00

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

Value 3
  Name:            EnableMCE
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            ExcludeFromKnownDlls
  Type:            REG_MULTI_SZ
  Data:

Value 5
  Name:            GlobalFlag
  Type:            REG_DWORD
  Data:            0

Value 6
  Name:            HeapDeCommitFreeBlockThreshold
  Type:            REG_DWORD
  Data:            0

Value 7
  Name:            HeapDeCommitTotalFreeThreshold
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            HeapSegmentCommit
  Type:            REG_DWORD
  Data:            0

Value 9
  Name:            HeapSegmentReserve
  Type:            REG_DWORD
  Data:            0

Value 10
  Name:            LicensedProcessors
  Type:            REG_DWORD
  Data:            0x4

Value 11
  Name:            ObjectDirectories
  Type:            REG_MULTI_SZ
  Data:            \Windows
                   \RPC Control

Value 12
  Name:            ProcessorControl
  Type:            REG_DWORD
  Data:            0x2

Value 13
  Name:            ProtectionMode
  Type:            REG_DWORD
  Data:            0

Value 14
  Name:            RegisteredProcessors
  Type:            REG_DWORD
  Data:            0x4

Value 15
  Name:            ResourceTimeoutCount
  Type:            REG_DWORD
  Data:            0x9e340
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session Manager\AppPatches
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\CWD
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\CWD\ff060102423da0000407108e0500
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\CWD\ff060102423da0000407108e0500\1
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Add1
  Type:            REG_BINARY
  Data:
00000000   02 15 40 a0 10 1e b8 23 - 00 8e d8 8b 0e 14 07 81  ..@....#........
00000010   e1 00 02 1f c3                                     .....

Value 1
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 1d 50 48 0c 55 8b ec - b8 00 00 9c 59 81 e1 00  ..PH.U......Y...
00000010   02 55 8b ec b8 00 00 e8 - e7 57 90 90 90           .U.......W...


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\MYST
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\MYST\ff060102423bab000407102e0600
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\MYST\ff060102423bab000407102e0600\1
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Add1
  Type:            REG_BINARY
  Data:
00000000   02 15 40 ab 10 1e b8 23 - 00 8e d8 8b 0e 14 07 81  ..@....#........
00000010   e1 00 02 1f c3                                     .....

Value 1
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 1d 50 49 0c 55 8b ec - b8 00 00 9c 59 81 e1 00  ..PI.U......Y...
00000010   02 55 8b ec b8 00 00 e8 - e7 61 90 90 90           .U.......a...


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\PALED40
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\PALED40\ff06010242003200407401b0100
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\PALED40\ff06010242003200407401b0100\1
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 07 b7 21 01 d8 0c                              ...!...


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\USA
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\USA\ff06010242059b00040710780600
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\USA\ff06010242059b00040710780600\1
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 1d 95 44 0c 55 8b ec - b8 00 00 9c 59 81 e1 00  ...D.U......Y...
00000010   02 55 8b ec b8 00 00 e8 - 67 56 90 90 90           .U......gV...

Value 1
  Name:            Change2
  Type:            REG_BINARY
  Data:
00000000   01 25 05 9b 10 00 00 00 - 00 00 00 00 00 00 00 00  .%..............
00000010   00 00 00 00 00 1e b8 23 - 00 8e d8 8b 0e 14 07 81  .......#........
00000020   e1 00 02 1f c3                                     .....

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB\ff060102ec353f00040780c81300
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB\ff060102ec353f00040780c81300\12
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 11 1b 03 06 81 3e ba - 31 34 03 81 3e ba 31 09  ......>.14..>.1.
00000010   03                                                 .

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB40016
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB40016\ff0702021401ee3e000407d0460e00
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB40016\ff0702021401ee3e000407d0460e00\16
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 11 6d 2a 06 81 3e 6e - 36 34 03 81 3e 6e 36 09  ..m*..>n64..>n6.
00000010   03                                                 .

Key Name:          SYSTEM\CurrentControlSet\Control\Session Manager\DOS Devices
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            AUX
  Type:            REG_SZ
  Data:            \DosDevices\COM1

Value 1
  Name:            MAILSLOT
  Type:            REG_SZ
  Data:            \Device\MailSlot

Value 2
  Name:            NUL
  Type:            REG_SZ
  Data:            \Device\Null
```

```
Value 3
  Name:           PIPE
  Type:           REG_SZ
  Data:           \Device\NamedPipe

Value 4
  Name:           PRN
  Type:           REG_SZ
  Data:           \DosDevices\LPT1

Value 5
  Name:           UNC
  Type:           REG_SZ
  Data:           \Device\Mup


Key Name:         SYSTEM\CurrentControlSet\Control\Session Manager\Environment
Class Name:       <NO CLASS>
Last Write Time:  11/3/98 - 1:43 PM
Value 0
  Name:           APPDIR
  Type:           REG_SZ
  Data:           C:\inetpub\wwwroot

Value 1
  Name:           ComSpec
  Type:           REG_EXPAND_SZ
  Data:           %SystemRoot%\system32\cmd.exe

Value 2
  Name:           HOME
  Type:           REG_SZ
  Data:           C:/

Value 3
  Name:           NTRESKIT
  Type:           REG_SZ
  Data:           C:\NTRESKIT

Value 4
  Name:           NUMBER_OF_PROCESSORS
  Type:           REG_SZ
  Data:           2

Value 5
  Name:           OS
  Type:           REG_SZ
  Data:           Windows_NT

Value 6
  Name:           Os2LibPath
  Type:           REG_EXPAND_SZ
  Data:           %SystemRoot%\system32\os2\dll;

Value 7
  Name:           Path
  Type:           REG_SZ
  Data:
C:\mksnt;C:\WINNT\system32;C:\WINNT;;C:\MSSQL\BINN;C:\TUXEDO\bin;C:\NTRESKIT

Value 8
  Name:           PROCESSOR_ARCHITECTURE
  Type:           REG_SZ
  Data:           x86

Value 9
  Name:           PROCESSOR_IDENTIFIER
  Type:           REG_SZ
  Data:           x86 Family 6 Model 5 Stepping 1, GenuineIntel

Value 10
  Name:           PROCESSOR_LEVEL
  Type:           REG_SZ
  Data:           6

Value 11
  Name:           PROCESSOR_REVISION
  Type:           REG_SZ
  Data:           0501

Value 12
  Name:           ROOTDIR
  Type:           REG_SZ
  Data:           C:/

Value 13
  Name:           SHELL
  Type:           REG_SZ
```

```
  Data:           C:/mksnt/sh.exe

Value 14
  Name:           TMCONTEXTS
  Type:           REG_SZ
  Data:           1

Value 15
  Name:           TMPDIR
  Type:           REG_SZ
  Data:           C:/TEMP

Value 16
  Name:           TUXCONFIG
  Type:           REG_SZ
  Data:           C:\inetpub\wwwroot\tuxconfig

Value 17
  Name:           TUXDIR
  Type:           REG_EXPAND_SZ
  Data:           C:\TUXEDO

Value 18
  Name:           windir
  Type:           REG_EXPAND_SZ
  Data:           %SystemRoot%


Key Name:         SYSTEM\CurrentControlSet\Control\Session Manager\Executive
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM
Value 0
  Name:           AdditionalCriticalWorkerThreads
  Type:           REG_DWORD
  Data:           0

Value 1
  Name:           AdditionalDelayedWorkerThreads
  Type:           REG_DWORD
  Data:           0

Value 2
  Name:           PriorityQuantumMatrix
  Type:           REG_BINARY
  Data:
00000000   f0 e0 0d f9 00 00 00 00 - a3 01 bd 01             ............


Key Name:         SYSTEM\CurrentControlSet\Control\Session
Manager\FileRenameOperations
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM

Key Name:         SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM
Value 0
  Name:           advapi32
  Type:           REG_SZ
  Data:           advapi32.dll

Value 1
  Name:           comdlg32
  Type:           REG_SZ
  Data:           comdlg32.dll

Value 2
  Name:           crtdll
  Type:           REG_SZ
  Data:           crtdll.dll

Value 3
  Name:           DllDirectory
  Type:           REG_EXPAND_SZ
  Data:           %SystemRoot%\system32

Value 4
  Name:           gdi32
  Type:           REG_SZ
  Data:           gdi32.dll

Value 5
  Name:           kernel32
  Type:           REG_SZ
  Data:           kernel32.dll

Value 6
  Name:           lz32
```

```
  Type:           REG_SZ
  Data:           lz32.dll

Value 7
  Name:           ole32
  Type:           REG_SZ
  Data:           ole32.dll

Value 8
  Name:           oleaut32
  Type:           REG_SZ
  Data:           oleaut32.dll

Value 9
  Name:           olecli32
  Type:           REG_SZ
  Data:           olecli32.dll

Value 10
  Name:           olecnv32
  Type:           REG_SZ
  Data:           olecnv32.dll

Value 11
  Name:           olesvr32
  Type:           REG_SZ
  Data:           olesvr32.dll

Value 12
  Name:           olethk32
  Type:           REG_SZ
  Data:           olethk32.dll

Value 13
  Name:           rpcrt4
  Type:           REG_SZ
  Data:           rpcrt4.dll

Value 14
  Name:           shell32
  Type:           REG_SZ
  Data:           shell32.dll

Value 15
  Name:           user32
  Type:           REG_SZ
  Data:           user32.dll

Value 16
  Name:           version
  Type:           REG_SZ
  Data:           version.dll


Key Name:         SYSTEM\CurrentControlSet\Control\Session Manager\Memory
Management
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM
Value 0
  Name:           ClearPageFileAtShutdown
  Type:           REG_DWORD
  Data:           0

Value 1
  Name:           DisablePagingExecutive
  Type:           REG_DWORD
  Data:           0

Value 2
  Name:           IoPageLockLimit
  Type:           REG_DWORD
  Data:           0

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

Value 4
  Name:           NonPagedPoolQuota
  Type:           REG_DWORD
  Data:           0

Value 5
  Name:           NonPagedPoolSize
  Type:           REG_DWORD
  Data:           0
```

```
Value 6
  Name:            PagedPoolQuota
  Type:            REG_DWORD
  Data:            0

Value 7
  Name:            PagedPoolSize
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            PagingFiles
  Type:            REG_MULTI_SZ
  Data:            C:\pagefile.sys 400 400


Value 9
  Name:            SecondLevelDataCache
  Type:            REG_DWORD
  Data:            0

Value 10
  Name:            SystemPages
  Type:            REG_DWORD
  Data:            0


Key Name:          SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Debug
  Type:            REG_EXPAND_SZ
  Data:

Value 1
  Name:            Kmode
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\win32k.sys

Value 2
  Name:            Optional
  Type:            REG_MULTI_SZ
  Data:            Os2
                   Posix

Value 3
  Name:            Os2
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\os2ss.exe

Value 4
  Name:            Posix
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\psxss.exe

Value 5
  Name:            Required
  Type:            REG_MULTI_SZ
  Data:            Debug
                   Windows


Value 6
  Name:            Windows
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows
SharedSection=1024,3072,1024 Windows=On SubSystemType=Windows
ServerDll=basesrv,1 ServerDll=winsrv:UserServerDllInitialization,3
ServerDll=winsrv:ConServerDllInitialization,2 ProfileControl=Off
MaxRequestThreads=16
```

Inetinfo

```
Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
Class Name:        <NO CLASS>
Last Write Time:   10/5/98 - 3:54 PM
Value 0
  Name:            BandwidthLevel
  Type:            REG_DWORD
  Data:            0xffffffff
```

```
Value 1
  Name:           ListenBackLog
  Type:           REG_DWORD
  Data:           0x1770

Value 2
  Name:           MaxPoolThreads
  Type:           REG_DWORD
  Data:           0x64

Value 3
  Name:           PoolThreadLimit
  Type:           REG_DWORD
  Data:           0x64

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


Key Name:         SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\Filter
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 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:  7/15/97 - 3:49 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:


Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Performance
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            Close
  Type:            REG_SZ
  Data:            CloseINFOPerformanceData

Value 1
  Name:            Collect
  Type:            REG_SZ
  Data:            CollectINFOPerformanceData

Value 2
  Name:            First Counter
  Type:            REG_DWORD
  Data:            0x738

Value 3
  Name:            First Help
  Type:            REG_DWORD
  Data:            0x739

Value 4
  Name:            Last Counter
  Type:            REG_DWORD
  Data:            0x756

Value 5
  Name:            Last Help
  Type:            REG_DWORD
  Data:            0x757

Value 6
  Name:            Library
  Type:            REG_SZ
  Data:            infoctrs.DLL

Value 7
  Name:            Open
  Type:            REG_SZ
  Data:            OpenINFOPerformanceData
```

Mssql

```
Key Name:          SOFTWARE\Microsoft\MSSQLServer
Class Name:        <NO CLASS>
Last Write Time:   12/8/97 - 12:05 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\Client
Class Name:        <NO CLASS>
Last Write Time:   12/8/97 - 12:06 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\Client\ConnectTo
Class Name:        <NO CLASS>
Last Write Time:   10/26/98 - 12:31 PM
Value 0
  Name:            DSQUERY
  Type:            REG_SZ
  Data:            DBMSSOCN


Key Name:          SOFTWARE\Microsoft\MSSQLServer\Client\DB-Lib
Class Name:        <NO CLASS>
Last Write Time:   9/2/98 - 1:24 PM
Value 0
  Name:            AutoAnsiToOem
  Type:            REG_SZ
  Data:            on

Value 1
  Name:            UseIntlSettings
  Type:            REG_SZ
  Data:            ON
```

```
Key Name:           SOFTWARE\Microsoft\MSSQLServer\ClientSetup
Class Name:         <NO CLASS>
Last Write Time:    12/8/97 - 12:06 PM
Value 0
  Name:             SQLPath
  Type:             REG_SZ
  Data:             C:\MSSQL
```

Tpcc

```
Key Name:           SOFTWARE\Microsoft\Tcpip
Class Name:         GenericClass
Last Write Time:    12/5/97 - 1:30 AM

Key Name:           SOFTWARE\Microsoft\Tcpip\CurrentVersion
Class Name:         GenericClass
Last Write Time:    12/5/97 - 1:30 AM
Value 0
  Name:             Description
  Type:             REG_SZ
  Data:             Transport Control Protocol/Internet Protocol. The default
wide area network protocol that provides communication across diverse
interconnected networks.

Value 1
  Name:             InstallDate
  Type:             REG_DWORD
  Data:             0x34883a1a

Value 2
  Name:             MajorVersion
  Type:             REG_DWORD
  Data:             0x4

Value 3
  Name:             MinorVersion
  Type:             REG_DWORD
  Data:             0

Value 4
  Name:             OperationsSupport
  Type:             REG_DWORD
  Data:             0x86

Value 5
  Name:             RefCount
  Type:             REG_DWORD
  Data:             0

Value 6
  Name:             Review
```

```
  Type:            REG_DWORD
  Data:            0x1

Value 7
  Name:            ServiceName
  Type:            REG_SZ
  Data:            Tcpip

Value 8
  Name:            SoftwareType
  Type:            REG_SZ
  Data:            transport

Value 9
  Name:            Title
  Type:            REG_SZ
  Data:            TCP/IP Protocol


Key Name:          SOFTWARE\Microsoft\Tcpip\CurrentVersion\NetRules
Class Name:        GenericClass
Last Write Time:   12/5/97 - 1:30 AM
Value 0
  Name:            bindable
  Type:            REG_MULTI_SZ
  Data:            tcpipService tcpipTransport non exclusive 100
                   tcpipTransport ndisDriver non non 100


Value 1
  Name:            bindform
  Type:            REG_SZ
  Data:            "Tcpip" yes yes container

Value 2
  Name:            class
  Type:            REG_MULTI_SZ
  Data:            tcpipTransport basic
                   tcpipService basic yes


Value 3
  Name:            InfName
  Type:            REG_SZ
  Data:            oemnxptc.inf

Value 4
  Name:            InfOption
  Type:            REG_SZ
  Data:            TC

Value 5
  Name:            type
  Type:            REG_SZ
  Data:            tcpip tcpipTransport

Value 6
  Name:            use
  Type:            REG_SZ
  Data:            transport none none
```

W3svc

```
Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM
Value 0
  Name:            DependOnGroup
  Type:            REG_MULTI_SZ
  Data:

Value 1
  Name:            DependOnService
  Type:            REG_MULTI_SZ
  Data:            RPCSS
                   NTLMSSP


Value 2
  Name:            DisplayName
  Type:            REG_SZ
  Data:            World Wide Web Publishing Service

Value 3
  Name:            ErrorControl
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            ImagePath
  Type:            REG_EXPAND_SZ
  Data:            C:\WINNT\System32\inetsrv\inetinfo.exe

Value 5
  Name:            ObjectName
  Type:            REG_SZ
  Data:            LocalSystem

Value 6
  Name:            Start
  Type:            REG_DWORD
  Data:            0x2

Value 7
  Name:            Type
  Type:            REG_DWORD
  Data:            0x20


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Enum
Class Name:        <NO CLASS>
Last Write Time:   11/3/98 - 1:43 PM
Value 0
  Name:            0
  Type:            REG_SZ
  Data:            Root\LEGACY_W3SVC\0000

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

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


Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\HTMLA
Class Name:        <NO CLASS>
Last Write Time:   7/15/97 - 3:49 PM

Key Name:          SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
Class Name:        <NO CLASS>
Last Write Time:   8/21/97 - 3:57 PM
Value 0
  Name:            AcceptExOutstanding
  Type:            REG_DWORD
  Data:            0x1770

Value 1
  Name:            AccessDeniedMessage
  Type:            REG_SZ
  Data:            Error: Access is Denied.

Value 2
  Name:            AdminEmail
  Type:            REG_SZ
```

```
  Data:           Admin@corp.com

Value 3
  Name:           AdminName
  Type:           REG_SZ
  Data:           Administrator

Value 4
  Name:           AnonymousUserName
  Type:           REG_SZ
  Data:           IUSR_BDR1_0000000000

Value 5
  Name:           Authorization
  Type:           REG_DWORD
  Data:           0x5

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

Value 7
  Name:           CheckForWAISDB
  Type:           REG_DWORD
  Data:           0

Value 8
  Name:           ConnectionTimeOut
  Type:           REG_DWORD
  Data:           0x2260

Value 9
  Name:           DebugFlags
  Type:           REG_DWORD
  Data:           0x8

Value 10
  Name:           Default Load File
  Type:           REG_SZ
  Data:           Default.htm

Value 11
  Name:           Dir Browse Control
  Type:           REG_DWORD
  Data:           0x4000001e

Value 12
  Name:           Filter DLLs
  Type:           REG_SZ
  Data:           C:\WINNT\System32\inetsrv\sspifilt.dll

Value 13
  Name:           GlobalExpire
  Type:           REG_DWORD
  Data:           0xffffffff

Value 14
  Name:           InstallPath
  Type:           REG_SZ
  Data:           C:\WINNT\System32\inetsrv

Value 15
  Name:           LogFileDirectory
  Type:           REG_EXPAND_SZ
  Data:           %SystemRoot%\System32\LogFiles

Value 16
  Name:           LogFileFormat
  Type:           REG_DWORD
  Data:           0

Value 17
  Name:           LogFilePeriod
  Type:           REG_DWORD
  Data:           0x1

Value 18
  Name:           LogFileTruncateSize
  Type:           REG_DWORD
  Data:           0x1388000

Value 19
  Name:           LogSqlDataSource
  Type:           REG_SZ
  Data:           HTTPLOG

Value 20
```

```
  Name:           LogSqlPassword
  Type:           REG_SZ
  Data:           sqllog

Value 21
  Name:           LogSqlTableName
  Type:           REG_SZ
  Data:           Internetlog

Value 22
  Name:           LogSqlUserName
  Type:           REG_SZ
  Data:           InternetAdmin

Value 23
  Name:           LogType
  Type:           REG_DWORD
  Data:           0

Value 24
  Name:           MajorVersion
  Type:           REG_DWORD
  Data:           0x2

Value 25
  Name:           MaxConnections
  Type:           REG_DWORD
  Data:           0x2000

Value 26
  Name:           MinorVersion
  Type:           REG_DWORD
  Data:           0

Value 27
  Name:           NTAuthenticationProviders
  Type:           REG_SZ
  Data:           NTLM

Value 28
  Name:           ScriptTimeout
  Type:           REG_DWORD
  Data:           0x384

Value 29
  Name:           SecurePort
  Type:           REG_DWORD
  Data:           0x1bb

Value 30
  Name:           ServerComment
  Type:           REG_SZ
  Data:

Value 31
  Name:           ServerSideIncludesEnabled
  Type:           REG_DWORD
  Data:           0x1

Value 32
  Name:           ServerSideIncludesExtension
  Type:           REG_SZ
  Data:           .stm


Key Name:         SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Script
Map
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 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:  7/15/97 - 3:49 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


Key Name:         SYSTEM\CurrentControlSet\Services\W3SVC\Performance
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM
Value 0
  Name:           Close
  Type:           REG_SZ
  Data:           CloseW3PerformanceData

Value 1
  Name:           Collect
  Type:           REG_SZ
  Data:           CollectW3PerformanceData

Value 2
  Name:           First Counter
  Type:           REG_DWORD
  Data:           0x758

Value 3
  Name:           First Help
  Type:           REG_DWORD
  Data:           0x759

Value 4
  Name:           Last Counter
  Type:           REG_DWORD
  Data:           0x790

Value 5
  Name:           Last Help
  Type:           REG_DWORD
  Data:           0x791

Value 6
  Name:           Library
  Type:           REG_SZ
  Data:           w3ctrs.DLL
```

```
Value 7
  Name:           Open
  Type:           REG_SZ
  Data:           OpenW3PerformanceData


Key Name:         SYSTEM\CurrentControlSet\Services\W3SVC\Security
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM
Value 0
  Name:           Security
  Type:           REG_BINARY
  Data:
00000000   01 00 14 80 c0 00 00 00 - cc 00 00 00 14 00 00 00  ................
00000010   34 00 00 00 02 00 20 00 - 01 00 00 00 02 80 18 00  4..... .........
00000020   ff 01 0f 00 01 01 00 00 - 00 00 00 01 00 00 00 00  ................
00000030   20 02 00 00 02 00 8c 00 - 05 00 00 00 00 00 18 00   ...............
00000040   8d 01 02 00 01 01 00 00 - 00 00 00 01 00 00 00 00  ................
00000050   72 00 76 00 00 00 1c 00 - fd 01 02 00 01 02 00 00  r.v.............
00000060   00 00 00 05 20 00 00 00 - 23 02 00 00 69 00 63 00  .... ...#...i.c.
00000070   00 00 1c 00 ff 01 0f 00 - 01 02 00 00 00 00 00 05  ................
00000080   20 00 00 00 20 02 00 00 - 69 00 63 00 00 00 1c 00   ... ...i.c.....
00000090   ff 01 0f 00 01 02 00 00 - 00 00 00 05 20 00 00 00  ............ ...
000000a0   25 02 00 00 69 00 63 00 - 00 00 18 00 fd 01 02 00  %...i.c.........
000000b0   01 01 00 00 00 00 00 05 - 12 00 00 00 25 02 00 00  ............%...
000000c0   01 01 00 00 00 00 00 05 - 12 00 00 00 01 01 00 00  ................
000000d0   00 00 00 05 12 00 00 00 -                          ........


Key Name:         SYSTEM\CurrentControlSet\Services\W3SVC\W3SAMP
Class Name:       <NO CLASS>
Last Write Time:  7/15/97 - 3:49 PM
```

C.10 Tuxedo Configuration

Client 1

```
**RESOURCES
IPCKEY          123456

DOMAINID tpcc
MASTER          tpcc
MAXACCESSERS    768
MAXSERVERS      256
MAXSERVICES     768
MAXGROUPS       256
MAXMACHINES     256
MODEL           SHM
LDBAL             N
SCANUNIT 60
BLOCKTIME       300
SANITYSCAN      300
DBBLWAIT  2
BBLQUERY 300


*MACHINES
DEFAULT:
APPDIR="c:\inetpub\wwwroot"
TUXCONFIG="c:\inetpub\wwwroot\tuxconfig"
TUXDIR="c:\tuxedo"

UWLC1   LMID=tpcc

*GROUPS
TPCCGROUP
LMID=tpcc       GRPNO=1  OPENINFO=NONE

*SERVERS
DEFAULT:
CLOPT="-A"
tux_server      SRVGRP=TPCCGROUP SRVID=1 RQADDR=tpcc REPLYQ=Y MIN=68

*SERVICES
NEW_ORDER
STOCK_LEVEL
PAYMENT
ORDER_STATUS
```

Client 2

```
*RESOURCES
IPCKEY          123456

DOMAINID tpcc
MASTER          tpcc
MAXACCESSERS    768
MAXSERVERS      256
MAXSERVICES     768
MAXGROUPS       256
MAXMACHINES     256
MODEL           SHM
LDBAL             N
SCANUNIT 60
BLOCKTIME       300
SANITYSCAN      300
DBBLWAIT  2
BBLQUERY 300


*MACHINES
DEFAULT:
APPDIR="c:\inetpub\wwwroot"
TUXCONFIG="c:\inetpub\wwwroot\tuxconfig"
TUXDIR="c:\tuxedo"

UWLC2   LMID=tpcc

*GROUPS
TPCCGROUP
LMID=tpcc       GRPNO=1  OPENINFO=NONE
```

```
*SERVERS
DEFAULT:
CLOPT="-A"

tux_server      SRVGRP=TPCCGROUP SRVID=1 RQADDR=tpcc REPLYQ=Y MIN=68

*SERVICES
NEW_ORDER
STOCK_LEVEL
PAYMENT
ORDER_STATUS
```

Client 3

```
*RESOURCES
IPCKEY          123456

DOMAINID tpcc
MASTER          tpcc
MAXACCESSERS    768
MAXSERVERS      256
MAXSERVICES     768
MAXGROUPS       256
MAXMACHINES     256
MODEL           SHM
LDBAL             N
SCANUNIT 60
BLOCKTIME       300
SANITYSCAN      300
DBBLWAIT  2
BBLQUERY 300


*MACHINES
DEFAULT:
APPDIR="c:\inetpub\wwwroot"
TUXCONFIG="c:\inetpub\wwwroot\tuxconfig"
TUXDIR="c:\tuxedo"

UWLC4    LMID=tpcc

*GROUPS
TPCCGROUP
LMID=tpcc       GRPNO=1  OPENINFO=NONE

*SERVERS
DEFAULT:
CLOPT="-A"

tux_server      SRVGRP=TPCCGROUP SRVID=1 RQADDR=tpcc REPLYQ=Y MIN=68

*SERVICES
NEW_ORDER
STOCK_LEVEL
PAYMENT
ORDER_STATUS
```

Client 4

```
*RESOURCES
IPCKEY          123456

DOMAINID tpcc
MASTER          tpcc
MAXACCESSERS    768
MAXSERVERS      256
MAXSERVICES     768
MAXGROUPS       256
MAXMACHINES     256
MODEL           SHM
LDBAL             N
SCANUNIT 60
BLOCKTIME       300
SANITYSCAN      300
DBBLWAIT  2
BBLQUERY 300


*MACHINES
DEFAULT:
APPDIR="c:\inetpub\wwwroot"
TUXCONFIG="c:\inetpub\wwwroot\tuxconfig"
TUXDIR="c:\tuxedo"
```

```
UWLC5    LMID=tpcc

*GROUPS
TPCCGROUP
LMID=tpcc        GRPNO=1  OPENINFO=NONE

*SERVERS
DEFAULT:
CLOPT="-A"

tux_server       SRVGRP=TPCCGROUP SRVID=1 RQADDR=tpcc REPLYQ=Y MIN=68

*SERVICES
NEW_ORDER
STOCK_LEVEL
PAYMENT
ORDER_STATUS
```

C.11 RTE Configuration parameters

```
Run Type: non batch
Run options: -u 18850
Environment:

_=/usr/bin/env
KINDS=HOST SERVER CHECKPOINT CLIENT DELIVERY DRIVER
NR_HOST=1
CHKPNT_INTERVAL2=29
CHECKPOINT_USER=tpcc
MANPATH=/usr/share/man/%L:/usr/share/man:/usr/contrib/man/%L:/usr/contrib/man:/
usr/local/man/%L:/usr/local/man:/opt/blinklink/share/man:/opt/ansic/share/man/%
L:/opt/ansic/share/man:/opt/langtools/share/man/%L:/opt/langtools/share/man:/op
t/dtcmgr/share/man:/opt/pd/share/man/%L:/opt/pd/share/man:/usr/omni/man
OS_MENU=0.10
EMON=\runemon
RPT_WINDOW_SIZE=30
SERVER_MONITOR=\ntreskit\monitor.exe
SERVER_SQL_OPTIONS=-x -c -T3502
CLIENT_SHUTDOWN=\ntreskit\shutdown.exe
RESULTS_NAME=e2
DRIVER_SPAWN_PERIOD=1
SERVER_SQLSTATS=true
SERVER_EMON_INTERVAL=100
KRATE1=360
STKL_THINK=5.15
VISUAL=/usr/bin/vi
CHECKPOINT_SQL_DIR=C:\mssql7
KRATE2=360
TERMKNOWN=yes
PATH=/home/web-tpcc/bin:/home/web-
tpcc/scripts:/usr/bin:/opt/ansic/bin:/usr/ccs/bin:/usr/contrib/bin:/opt/nettlad
m/bin:/usr/bin/X11:/usr/contrib/bin/X11:/opt/langtools/bin:/opt/dtcmgr/sbin:/op
t/pd/bin:/opt/upgrade/bin:/sbin:/usr/sbin:/usr/sbin:.:/usr/local/bin:/usr/local
/scripts:/usr/bin/X11:/usr/local/bin/X11:/usr/contrib/condor/bin:/usr/lib/SoftW
indows/bin
RUNENV=/home/web-tpcc/RUNENV
DELIVERYS=uwlc1 uwlc1b uwlc2 uwlc2b uwlc4 uwlc4b uwlc5 uwlc5c
NUMBER=1
EMONDUR=420
RESULT_DIR=/home/web-tpcc/results/e2.16
DELIVERY_TYPE=iis-web-delivery
NR_DRIVER=16
SERVER_DB_GAMINIT=false
CHECKPOINT_TEMP_NAME=/tmp/checkpoint.15332
SERVER_RES_KIT=\ntreskit
OUTPUT_DIR=/home/web-tpcc/results
CLAST_CONST_C=190
COLUMNS=75
CHECKPOINT_TOUCH=\mksnt\touch.exe
SERVER_USER=tpcc
CLIENT_TUX_DIR=c:\tuxedo
BATCH_TPCC=false
NR_CHECKPOINT=1
TPCC_SERVICE=80
CHKPNT_SQL_DIR=C:\mssql7
NR_CLIENT=8
DRIVER_PATH=/home/web-tpcc/bin/hpux-web-driver
COMM_ADJUST_PMT=0.10
CLIENT_CONFINFO=\tools\bin\confinfo.bat
SERVER_EMON_FIRST=300
ROUTINE=DRIVER_finish
CLIENT_RTIME_NAME=\temp\rtime.exe
CHECKPOINTS=uwlc1
CHECKPOINT=uwlc1 uwlc1b uwlc2 uwlc2b uwlc4 uwlc4b uwlc5 uwlc5c
DVRY_THINK=5.15
SERVER_TYPE=mssqlserver.70
SERVER_SC=\ntreskit\sc.exe
count=1
```

```
THIS_TYPE=hpux-web-driver
CHECKPOINT_LOG_PATH=\temp\checkpoint.log
SERVER_SQLISQL=\mssql7\binn\isql
DELIVERY_USER=tpcc
COMM_ADJUST_NEWO=0.10
CLIENT_RTIME_FILE=rtime.exe
EDITOR=/usr/bin/vi
CHECKPOINT_LOG_FILE=checkpoint.log
KERNRATE=\runkrate
NEWO_THINK=12.12
STKL_MENU=0.10
LOGNAME=web-tpcc
DRIVER_AUTO=false
DELIVERY_DELILOG=delilog
TESTENV=/tmp/RUNENV.15332
CHKPNTDUR=780
MAIL=/usr/mail/web-tpcc
SECONDS=10
CLIENTS=uwlc1 uwlc1b uwlc2 uwlc2b uwlc4 uwlc4b uwlc5 uwlc5c
CLIENT_PATH=/home/web-tpcc/bin/iis-web-tux
HOST_AUTO=false
POST_STEPS=shutdown stop_performance cleanup audit finish
PMT_MENU=0.10
DRIVER_PROG=driver.web
ERASE=^H
PS1=web-tpcc@uxr9%
INTERRUPTED=false
HOST_PATH=/home/web-tpcc/bin/hpux-driver-host
HOST=uxr9
TRANS_TIME=130
RUN_ID=16
CLIENT_AUTO=false
NR_SERVER=1
NR_DELIVERY=8
RUN_ID_FILE=/home/web-tpcc/run_id
CHECKPOINT_TYPE=mssqlserver-checkpoint
SERVER_RTIME_NAME=\temp\rtime.exe
CLIENT=uwlc1 uwlc1b uwlc2 uwlc2b uwlc4 uwlc4b uwlc5 uwlc5c
SKIP_IF_INTERRUPTED=true
SERVER_COLLECT_KERNRATE=false
SERVER_LOG_PATH=\mssql7\log\errorlog
RESULTS_ROOT=/home/web-tpcc/results
CONFINFO=\tools\bin\confinfo.bat
DRIVER=uxr8 uxr2 uxr1b uxr2b uxr3 uxr4 uxr3b uxr4b uxr5 uxr6 uxr5b uxr6b uxr7
uxr10 uxr1c uxr10c
SERVER_AUTO=false
DRIVER_WAIT_FIFO=prospect_wait
COMM_ADJUST_STKL=0.10
THIS_FILE=/home/web-tpcc/bin/hpux-web-driver/routines
DVRY_MENU=0.10
CHECKPOINT_SCRIPT=\temp\checkpoint.ksh
SERVER_RTIME_FILE=rtime.exe
SERVER_PATH=/home/web-tpcc/bin/mssqlserver.70
OUTPUT_PATH=/home/web-tpcc/results/output.16
SYSTEM=uxr9
DRIVERS=uxr8 uxr2 uxr1b uxr2b uxr3 uxr4 uxr3b uxr4b uxr5 uxr6 uxr5b uxr6b uxr7
uxr10 uxr1c uxr10c
PERFORMANCE_DETAIL=0
THIS_PATH=/home/web-tpcc/bin/hpux-web-driver
OUTPUT_FILE=output
SERVERS=prf_sut7
SERVER_KERNRATE_ARGS=1800 600 -K -s 360
GPROFDIR=.
DRIVER_TYPE=hpux-web-driver
BINDIR=/home/web-tpcc/bin
PREV_DIR=/home/web-tpcc
COUNT_USERS_TIME=75
CLIENT_IIS_PORT=80
DISPLAY=uxr9:0.0
COMM_ADJUST_DVRY=0.10
CHECKPOINT_STOP_FILE=/temp/checkpoint.stop
SERVER_PMW_NAME=\temp\tpcc-server.pmw
DRIVER_PROSPECT=/home/web-tpcc/bin/hpux-web-driver/prospect.B.10.20
NEWO_KEY=18.01
STKL_KEY=2.01
TRANS_TYPE=0
CHECKPOINT_PATH=/home/web-tpcc/bin/mssqlserver-checkpoint
SHELL=/usr/bin/ksh
DB_SIZE=1885
CID_CONST_C=498
CLIENT_TUX_APP_DIR=c:\inetpub\wwwroot
SERVER_SQLSERVR=\mssql7\binn\sqlservr
NEWO_MENU=0.10
CONFIGDIR=/home/web-tpcc/config
HOST_TYPE=hpux-driver-host
DELIVERY_DELISRV_PATH=\inetpub\wwwroot\delisrv.exe
HISTSIZE=50000
```

```
SWINHOME=/usr/lib/SoftWindows
PRE_STEPS=initialize startup start_performance
SERVER_COLLECT_EMON=false
SERVER_PMW_FILE=tpcc-server
CHECKPOINT_SQLISQL=C:\mssql7\binn\isql.exe
ARGS=-u 18850
TRANS_NUM=13000000
CHECKPOINT_AUTO=false
OUTPUT_LEVEL=3
HOME=/home/web-tpcc
DELIVERY=uwlc1 uwlc1b uwlc2 uwlc2b uwlc4 uwlc4b uwlc5 uwlc5c
CLIENT_TYPE=iis-web-tux
CLIENT_USER=tpcc
SERVER=prf_sut7
OS_THINK=10.15
IID_CONST_C=3415
MAILER=/usr/bin/elm
AUDIT=false
HOSTS=uxr9
TERM=hpterm
DVRY_KEY=2.01
HOST_USER=web-tpcc
CLIENT_MONITOR=\ntreskit\monitor.exe
PWD=/home/web-tpcc/results/e2.16
BASE_SHMEM_KEY=55555
CLIENT_PERF_LOG_NAME=\temp\tpcc-client.log
OS_KEY=2.01
COMM_ADJUST_ORDS=0.10
TZ=PST8PDT
PMT_KEY=3.01
CLIENT_PMW_FILE=tpcc-client
CLIENT_RES_KIT=\ntreskit
DRIVER_USER=web-tpcc
SERVER_FSPERF_LOG=\temp\fsperf.txt
ENV=/home/web-tpcc/.kshrc
DELIVERY_AUTO=false
PMT_THINK=12.15
SERVER_PERF_LOG_NAME=\temp\tpcc-server.log
SERVER_CACHING=false
CHKPNT_INTERVAL=20
DRIVER_SPAWN_RATE=1
DELIVERY_DELILOG_PATH=\inetpub\wwwroot\delilog
DELIVERY_PATH=/home/web-tpcc/bin/iis-web-delivery
CLIENT_PMW_NAME=\temp\tpcc-client.pmw
CLIENT_SC=\ntreskit\sc.exe
LINES=48
START_DIR=/home/web-tpcc
step=initialize
SERVER_SQL_DIR=\mssql7
MAILCHECK=300
A__z=-5MAILCHECK
```

Appendix D Disk Storage

TPC-C 180 Day Space Requirements						
Warehouses	**1885**				**TpmC**	**23143**
Table	**Rows**	**Data KB**	**Index KB**	**Extra 5% KB**	**8hr Space**	**Total Space KB**
Warehouse	1,885	208	56	13		277
District	18,850	2,096	72	108		2276
Customer	56,550,000	41,127,280	2,641,176	2,188,423		45956879
History	56,550,000	3,141,752	-		843,409	3141752
NewOrder	16,965,000	268,224	776			269000
Orders	56,550,000	1,733,336	957,456		5,085,220	2690792
OrderLine	497,998,198	35,343,696	88,064		8,352,799	35431760
Item	100,000	9,528	96	481		10105
Stock	188,500,000	60,320,000	135,216	3,022,761		63477977
Total		141,946,120	3,822,912	5,211,786	14,281,428	150,980,818

DB File Group	**Count**	**Size MB**	**MB Allocated**	**MB Loaded+5%**	**Total for 8 hrs**
Master,model,msdb	1	30	30	30	30
mssql70_tpcc_root	1	8	8	8	8
Mssql70_CS_FG	8	128,000	127,000	106,870	106,870
Mssql70_misc_FG	8	65,000	64,000	40,572	54,519
			191,038	147,480	161,427

	MB	
Dynamic Space	39,276	Sum of Data for Order, Orderline and History
Static Space	108,166	Sum of Data+Index+5%-Dynamic Space
Free Space	43,596	Total Allocated Spac - (Dynamic + Static Space)
Daily Growth	7,715	(Dynamic Space/(W*62.5))*tpmc
Daily Spread	32,023	(Free Space -1.5*Daily Growth) Zero Assumed

180 Day Space MB	1,496,934					
180 Day Space GB	**1,461.85**	**GB**				
Log Size	48,000	MB				
KB Per New Order	5.3590	KB				
8 hr log MB	58,136	MB				
8 hr log GB	**56.7730**	**GB**				

Space Usage	**GB Needed**	**Disks Priced**	**GB/disk**	**GB Priced**		
180 Day Space DB	1,461.85	168	8.47	1423.56	HP Hot-swap 9Gbyte 10krpm SCSI	
		24	4.24	101.70	HP Hot-swap 4Gbyte 10krpm SCSI	
Total DB		**192**		**1525.26**	**GB**	
8-hr log + mirror	113.5459	16	8.47	135.58	GB	
OS, Swap	6	1	8.47	8.4736	GB	
Total Storage	**1,581.40**	**GB**		**1,669.31**	**GB**	

Appendix E Quotations

E.1 Software House International

Software House International
Pricing Proposal

Quotation #MO-981210-57576
12/10/98

Hewlett Packard
Kuppuswamy Sivakumar
Quote good for Ninety Days

Phone: Fax: 447-8001

SHI Account Exec: Matthew O. Martin
Telephone : (408) 922-1106
Fax : (408) 526-1222

Reference:

Product	Part #	Qty	List	Your Price	Total
Windows NT Server Ent	Z75092	1		$3330.00	$3,330.00
Netserver LXR8000 P2400	D5021A	1		$11055.00	$11,055.00
Warranty Upgrade	H5516A	1		$1790.00	$1,790.00
Warranty Upgrade	H5519A	1		$659.00	$659.00
Total					$16,834.00

Additional Comments:

DEC-22-1998 10:51 SOFTWARE HOUSE INTL 408 232 2585 P.02/03

Software House International
Pricing Proposal

Quotation #MO-981208-57817
12/08/98

Hewlett Packard
Kuppuswamy Sivakumar
Quote Good For Ninety Days

,
Phone: **Fax: 447-8001**

SHI Account Exec: Matthew O. Martin
Telephone : (800) 52 - SOFTWare
Fax : (908) 805 - 0818

Reference:

Product	Part #	Qty	List	Your Price	Total
1GB Dimm Kit	D5114A	4		$4280.00	$17,120.00
RMKT 17in Ultra Monitor	D2818A	1		$213.00	$213.00
Netserver minidin	D4950B	1		$79.00	$79.00
9.1GB 10K Hot Swap	D5107A	1		$1022.00	$1,022.00
NetServer 10/100 Card	D5013A	1		$82.00	$82.00
Keyboard Rack Mount	E7714A	1		$150.00	$150.00
2.0 Meter Rack	J1497B	3		$1460.00	$4,380.00
HP Power Distribution	E5929A	5		$234.00	$1,170.00
HP 19in Plain Shelf	E3666A	1		$61.00	$61.00
APC Smart UPS 3000NS	588293	3		$1728.00	$5,184.00
Surestore Dat24i	C1555C	1		$960.00	$960.00
NetServer Rack Storage 8	D4902A	29		$2080.00	$60,320.00
4.2gb 10k Hot Swap	D4903A	27		$657.00	$17,739.00
9GB 10K Hot Swap	D5019A	203		$1030.00	$209,090.00
NetRaid-3Si Scsi	D5955A	8		$2075.00	$16,600.00
Total					**$334,170.00**

Additional Comments:

DEC-22-1998 10:50 SOFTWARE HOUSE INTL 408 232 2585 P.01/02

Software House International
Pricing Proposal

Quotation #MO-981208-64542
12/08/98

Hewlett Packard
Kuppuswamy Sivakumar
Quote Good For Ninety Days

Phone: Fax: 447-8001

SHI Account Exec: Matthew O. Martin
Telephone : (408) 922-1106
Fax : (408) 526-1222

Reference:

Product	Part #	Qty	List	Your Price	Total
Mylex DACSXI EDAD	Z75091	5		$3569.00	$17,845.00
HP SCSI Cable	D3637C	18		$82.00	$1,476.00
HP SCSI Cable	D3638C	13		$82.00	$1,066.00
HP SCSI Cable	C2911A	4		$122.00	$488.00
HP Storage System 6	D3604B	2		$997.00	$1,994.00
MKS Toolkit	323795			$258.00	$258.00
Netserver LC3	D6126T	4		$2455.00	$9,820.00
LC 3 P2400	D6092A4	4		$1138.00	$4,552.00
128MB Dimm	D5096A	16		$445.00	$7,120.00
NetServer 10/100	D5013A	4		$82.00	$328.00
RMKT 17in Display	D2618A	4		$213.00	$852.00
NT Server	Z75093	4		$568.00	$2,272.00
Visual C++	716856	1		$449.00	$449.00
Procurve	J4122A	2		$1189.00	$2,378.00
8port 10BT + 1 BNC	Z788822	2595		$28.75	$74,606.25
Total					$126,684.25

Additional Comments:

DEC-11-1998 16:13 SOFTWARE HOUSE INTL 408 232 2585 P.02/06

E.2 Microsoft

One Microsoft Way
Redmond, WA 98052-6399
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

December 10, 1998

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

via FAX # 408-873-5080

Dear Larry,

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

Microsoft SQL Server, Enterprise Edition 7.0, unlimited user license	$28999
Microsoft Windows NT Server, Enterprise Edition 4.0, incl 25 CALs	$3999
Windows NT Server 4.0, incl 5 CALs	$809
Visual C++ Professional 5.0	$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
Applications Marketing

One Microsoft Way
Redmond, WA 98052-6399
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

December 10, 1998

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

via FAX # 408-873-5080

Dear Larry,

Microsoft has received your request for permission to disclose the results of TPC-C benchmark tests conducted by HP with Microsoft SQL Server 7.0, Enterprise Edition on the following system:

HP NetServer LXr 8000, 4 processors, Pentium II Xeon, 450 MHz
Test Results: 23000 tpmC @ $29/tpmC approx.

Microsoft hereby grants HP permission to disclose these results to third parties and acknowledges that HP has formally requested permission to do so in accordance with the license agreement for Microsoft SQL Server 7.0, Enterprise Edition software.

Best regards,

Sid Arora
Product Manager, Microsoft SQL Server
Applications Marketing

E.3 BEA

ENTERPRISE MIDDLEWARE SOLUTIONS

December 8, 1998

Mr. Neil MacDonald
Hewlett Packard
Cupertino, CA

Dear Mr. MacDonald

Per your request I am enclosing the pricing information regarding TUXEDO 6.x that you requested. This pricing applies to Tuxedo 6.1, 6.2, 6.3 and 6.4. Please note that Tuxedo 6.4 is our most recent version of Tuxedo but that all 6.x releases are generally available. Core functionality services pricing is appropriate for your activities. As per the table below, HP server systems are classified in one of 5 tiers based on CPU type and capacity. This quote is valid for 90 days from the date of this letter.

Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.x. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees.

BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers (Class 1and Class 2)	Unlimited	$3,000.00	$450.00	$660.00
Tier 2 -- PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations (class 3)	Unlimited	$12,000.00	$1,800.00	$2,640.00
Tier 3 -- Midrange Multiprocessors, up to 8 CPUs per system capacity (Class 4 and 5)	Unlimited	$30,000.00	$4,500.00	$6,600.00
Tier 4 -- Large (more than 8, less than 32 CPUs) and Mainframe Systems (Class 6)	Unlimited	$100,000.00	$15,000.00	$22,000.00

BEA SYSTEMS, INC
2317 NO. FIRST STREET
SAN JOSE, CA 95131

FAX: 408-570-8901
PHONE: 408-570-8000

Tier 5 -- Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$37,500.00	$55,000.00

HP 9000 server tier classifications

Platform	Tier 1	Tier 1	Tier 2	Tier 3	Tier 3	Tier 4	Tier 5
Hewlett-Packard	Uni-processor Workstations	9000/E25 9000/E35 9000/E45 9000/E55 9000/G30 9000/G40	9000/G50 9000/G60 Multi-Processor Workstations 9000/D200, 210 220,230	9000/H20 9000/H30 9000/H40 9000/H50 9000/I30 9000/I40 9000/K1XX 9000/D310, 320,330 9000/D250/60	9000/I50,60 9000/H60 9000/G70 9000/H70 9000/I70 9000/K2XX 9000/K3XX 9000/K4XX 9000/K5XX 9000/D350/60/70	9000/T500, T600 1-16 CPUs	9000/V series all models

Intel based server tier classifications:

Platform	Operating System	Tier 1	Tier 1	Tier 2	Tier 3	Tier 3
Intel Pentium/ Pentium Pro PCs	Interactive R3.2 ESIX SVR 4.0 SCO UNIX 3.2.2 and 3.2.4 SCO ODT 2.x,3.x Solaris x86 2.X UnixWare, Windows NT 3.5/4.0	All 386/486 PCs are Class 1	ALL Pentium and Pentium Pro PCs with 1 or 2 CPUs capacity are Tier 1	ALL Pentium and Pentium Pro PCs with 3 or 4 CPUs capacity are Tier 2		ALL Pentium and Pentium Pro PCs with 5,6,7, or 8 CPUs are Tier 3

Very Truly Yours,

Lewis D. Brentano,
Director, Market Planning

E.4 APC

E.4 APC

AMERICAN POWER CONVERSION

American Power Conversion
132 Fairgrounds Rd.
West Kingston, RI 02892

FROM:Tracey Morton

FAX#: +1-401-789-3710
PHONE#: +1-401-789-5735

Price Quote for the 5 Year Power Plan Warranty

SU3000NS $396.00

American Power Conversion
Customer Service
Tracey Morton

E.5 AMI

JAN-18-1999

January 18, 1999

Spencer Frink
HP
Santa Clara, CA

Via FAX #: 408-447-8001

Dear Mr. Frink:

This is to notify you that American Megatrends, Inc. (AMI) will make available for general purchase AMI's Series 438-H Super High Performance RAID Controller with AMI's released Windows NT Monolithic driver v. I0110 and firmware v. H0330 or higher. This can be ordered directly from AMI. The part number is 4383508116 and price per unit is $2,075, (includes 16mb EDO memory and battery back up). This part is available now, and includes a 3 year limited warranty. An optional 2 year extended warranty can be purchased for the amount of $62 per unit.

Sincerely,

Stephen Bignault
RAID OEM Account Manager

6145-F Northbelt Parkway, Norcross, GA 600371-2976; Main: 770-246-8600; Direct: 770-246-8632; Fax: 770-246-8657

TOTAL P.01