NEC

TPC Benchmark™ C
Full Disclosure Report

NEC Express5800 HX4500 (4 SMP)

Using Microsoft SQL Server,Enterprise Edition 7.0
and Microsoft Windows NT Server,Enterprise Edition 4.0

First Edition
Submitted for Review
August 5, 1998

NEC, the Sponsors of this benchmark test, believe that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. The Sponsor assumes no responsibility for any errors that may appear in this document. The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, The Sponsor provides no warranty of the pricing information in this document.

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

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

Printed in USA, 1998

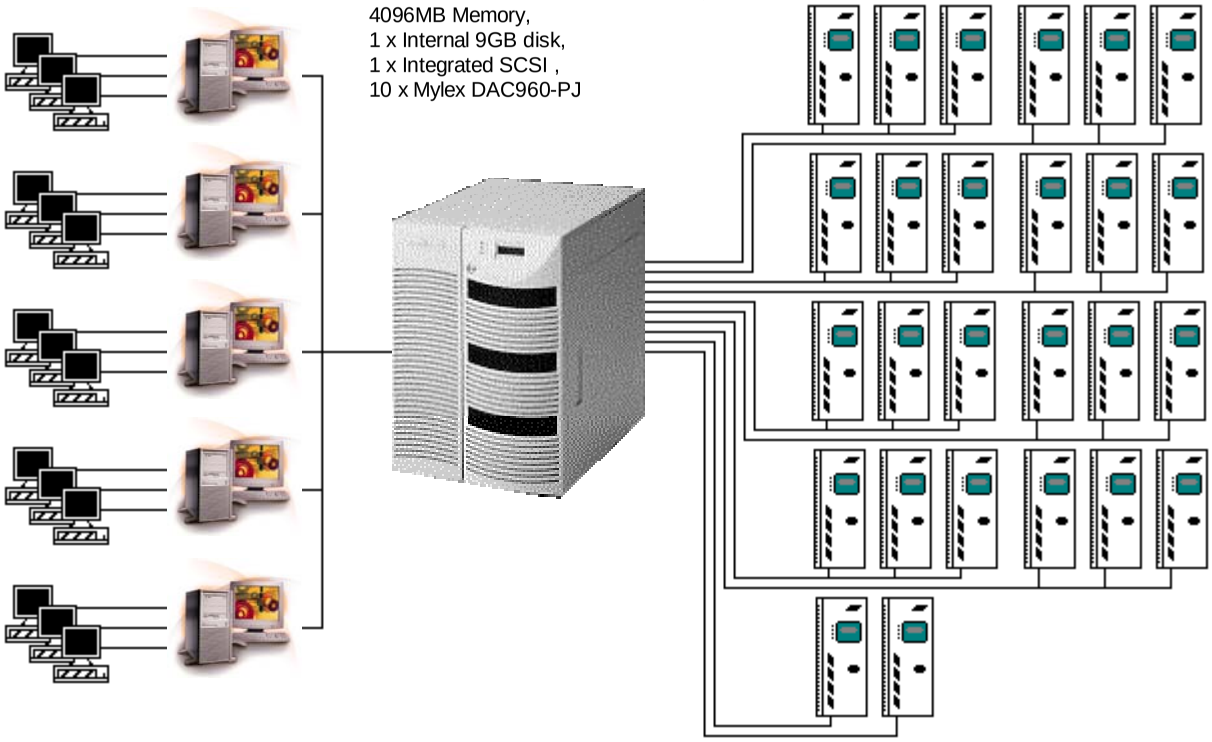
NEC	Express5800 HX4500 C/S		TPC-C Rev. 3.3 Reported Date August 5, 1998
Total System Cost	**TPC-C Throughput**	**Price/Performance**	**Availability Date**
$614,908	**18,322.67 tpmC**	**$33.55 per tpmC**	**December**

Processors	Database Manager	Operating System	Other Software	Number of Users
4 x Pentium ® II Xeon 400MHz 1MB L2 cache	Microsoft SQL Server 7.0 Enterprise Edition	Microsoft Windows NT4.0 Enterprise Edition	Microsoft IIS 3.0 BEA Tuxedo 6.3CFS Microsoft VC++	**14,780**

Total of 14,780 PCs

Server : Express5800 HX4500

4 x Pentium ® II Xeon 400MHz/1MB,
4096MB Memory,
1 x Internal 9GB disk,
1 x Integrated SCSI ,
10 x Mylex DAC960-PJ



5 x PowerMate Professional 9000-

333SP4M64N4

Storage: Disk Expansion Box
204 x 9GB disks

System Component	Server		Each Client	
Processors **Cache**	4	Pentium ® II Xeon 400MHz 1MB L2 Cache	2	Pentium ® II 333MHz 512KB
Memory	1	4096MB	1	512MB
Disk Controllers	10 1	Mylex DAC960-PJ Integrated SCSI	1	ULTRA WIDE SCSI
Disk Drives	205	9GB (8.47GB usable)	1	4.5GB
Total Storage		1736GB		
Others	1 1	CD-ROM Drive DAT Drive	1	CD-ROM Drive

NEC	NEC Express 5800 HX4500 C/S					TPC-C REV 3.3 Report Date: August 5, 1998	
Description	**Part Number**	**Third Party Brand**	**Pricing**	**Unit Price**	**Qty**	**Extended Price**	**5-year Maint. Price**
Server Hardware							
HX4500 system, 4xPentium II Xeon 400MHz/1MB , 4GB ECC Memory, integrated wide ultra SCSI, 3x4 drive hot swap disk bays, 3x420W power supply, 32x CD-ROM, keyboard, mouse	850127700	NEC	1	37,140	1	37,140	8,914
12/24GB SCSI DDS-3 4mm DAT Drive	203117	NEC	1	1,099	1	1,099	0
9GB 7,200 HDD	203283	NEC	1	899	1	899	0
NEC 15" Multisync Monitor	JC-1576VMA	NEC	1	299	1	299	72
Disk Subsystem							
Mylex DAC960-PJ 3 channel w/ 8MB EDO (+2spares)	DACPJ-3-8E-MY1	Mylex	4	1,845	12	22,140	500
Mylex Inteligent Battery Backup Unit (+2spairs)	DBBPG-MYL	Mylex	4	345	4	1,380	0
ST8000 DEU Tower Model	203269	NEC	1	3,999	26	103,974	24,954
9GB 10Krpm Hard Disk Drives	203350	NEC	1	999	204	203,796	0
68-pin VHD SCSI-3 data cable	203277	NEC	1	149	26	3,874	0
68-pin WIDE SCSI to VHD SCSI-3 data cable	203273	NEC	1	149	8	1,192	0
Single Bus Option Module	203330	NEC	1	89	26	2,314	0
					Subtotal	378,107	34,439
Server Software							
Microsoft Windows NT Server,Enterprise Edition 4.0		Microsoft	2	3,999	1	3,999	0
Microsoft SQL Server,Enterprise Edition 7.0 Unlimited		Microsoft	2	28,999	1	28,999	10,475
					Subtotal	32,998	10,475
Client Hardware							
PowerMate Professional 9000-333SP4M64N4 2 x Pentium II 333MHz, 512MB, 4.3GB HDD CD-ROM, KB,Mouse, Intel Pro 100	P9006P4M64N4	NEC	1	4,798	5	23,990	5,758
EtherExpress Pro/100+ NIC	203254	NEC	1	79	15	1,185	0
NEC 15" Multisync Monitor	JC-1576VMA	NEC	1	299	5	1,495	72
					Subtotal	26,670	5,830
Client Software							
Microsoft Windows NT Server 4.0 w/5 cal		Microsoft	2	809	5	4,045	0
Visual C++ Professional 5.0		Microsoft	2	499	1	499	0
Tuxedo 6.3 Core functionality Services for NT		BEA	3	3,000	5	15,000	11,250
					Subtotal	19,544	11,250
User Connectivity							
Linksys 16-port 100Mbps HUB (+2 spares)	FEHUB16W	Linksys	5	240	3	720	0
Linksys 20-prt 10Mbps HUB (+10% spares)	EW20HUB	Linksys	5	115	825	94,875	0
					Subtotal	95,595	0
					TOTAL	552,914	61,994

Notes:

All Microsoft maintenance is covered by the maintenance costs of Microsoft SQL Server

Pricing: 1-NEC 2-Microsoft 3-BEA 4-Mylex 5-BuyComp LLC

Linksys offers standard with 5-year warranty

Audited by Francois Raab of Information Paradigm, Inc

Five-Year Cost of Ownership:	**$614,908**
tpmC Rating:	**18322.67**
$ / tpmC:	**33.55**

Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflects 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

MQTh, Computed Maximum Qualified Throughput			18322.67 tpmC
% throughput difference ,reported & reproducibility runs			0.17 %
Response Times(in seconds)	**90%**	**Average**	**Maximum**
New-Order	0.70	0.46	17.01
Payment	0.56	0.32	9.35
Stock-Level	2.92	2.40	8.28
Delivery(interactive portion)	0.30	0.29	0.57
Delivery(deferred portion)	0.89	0.58	17.63
Order-status	0.63	0.40	11.78
Menu	0.31	0.20	1.02
Response time delay added for emulated components			0.1
Transaction Mix , in percent of total transaction			
New-Order			44.87%
Payment			43.01%
Order-status			4.03%
Delivery			4.06%
Stock-level			4.02%

Keying/Think Times (in seconds)	**Min.**		**Average**		**Max**	
New-Order	18.00	0.00	18.01	12.05	18.03	120.71
Payment	3.00	0.00	3.01	12.08	3.03	120.71
Stock-Level	2.00	0.00	2.01	5.07	2.03	48.46
Delivery	2.00	0.00	2.01	5.08	2.02	50.70
Order-status	2.00	0.00	2.01	10.08	2.03	100.70

Test Duration	
Ramp-up time	52 minutes
Measurement interval	30 minutes
Number of checkpoints	1
Checkpoint interval	30 minutes
Number of transactions (all types) completed in measurement interval	1,224,984

Abstract

This report documents the compliance of NEC Corporation's TPC Benchmark™ C tests on the NEC Express 5800/HX4500 client/server system with version 3.3.3 of the TPC Benchmark C Standard Specification. 5 Clients (NEC PowerMate Professional 9000-333SP4M64N4) were used as the front-end clients.

The operating system and the DBMS used on the server were Microsoft Windows NT,Enterprise Edition 4.0 and Microsoft SQL Server,Enterprise Edition 7.0. The operating system on the clients was Microsoft Windows NT Server 4.0(SP3). Those clients ran Microsoft's IIS server 3.0 and Tuxedo 6.3 CFS for Windows NT.

Two standard metrics, transaction-per-minute-C(tpmC) and price per tpmC($/tpmC) are reported, in accordance with the TPC Benchmark™ C Standard. The independent auditor's report by Francois Raab appears at the end of this report.

TPC Benchmark™ C Metrics

The standard TPC Benchmark™ C metrics, tpmC (transactions per minute), price per tpmC (five year capital cost per measured tpmC) are reported.

System	SW	Total System Cost	tpmC	$ per tpmC	Availability Date
NEC Express5800 HX4500	Microsoft Windows NT, Enterprise Edition 4.0 Microsoft SQL Server, Enterprise Edition 7.0 Tuxedo 6.3 CFS	$614,908	18322.67	$33.55	December 29, 1998

Standard and Executive Summary Statements

The following pages contain executive summary of results for this benchmark.

Auditor

The benchmark configuration, environment and methodology were audited by Francois Raab of Information Paradigm,Inc. to verify compliance with the relevant TPC specifications.

Preface

The TPC Benchmark™ C was developed by the Transaction Processing Performance Council (TPC). The TPC was founded to define transaction processing benchmarks and to disseminate objective, verifiable performance data to the industry. This full disclosure report is based on the TPC Benchmark™ C Standard Specifications Version 3.3.3.

TPC Benchmark™ C Overview

The TPC describes this benchmark in Clause 0.1 of the specifications as follows:

TPC Benchmark™ C is an On Line Transaction Processing (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.

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

Document Structure

This TPC Benchmark™ C Full Disclosure Report is organized as follows:

- The main body of the document lists each item in Clause 8 of the TPC-C Standard and explains how each requirement is satisfied.
- Appendix A contains the source code of the TPC-C application code used to implement the TPC-C transactions.
- Appendix B contains the database definition and population code used in the tests.
- Appendix C contains the tunable parameters used in the TPC-C tests.
- Appendix D contains space calculation table.
- Appendix E contains third-party price quotations.

TPC Benchmark™ C Full Disclosure

The TPC Benchmark™ C Standard Specification requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 8. This report is intended to satisfy the Standard's requirement for full disclosure. It documents the compliance of the benchmark tests with each item listed in Clause 8 of the TPC Benchmark™ C Standard Specification.

In the Standard Specification, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the Standard Specification, printed in italic type. The plain text that follows explains how the tests comply with the TPC Benchmark™ C requirement. In sections where Clause 8 requires extensive listings, the section refers to the appropriate appendix at the end of this report.

General Items

Order and titles

The order and titles of sections in the Test Sponsor's Full Disclosure Report must correspond with the order and titles of for TPC-C standard specification. The intent is to make it as easy as possible for readers to compare and contrast material in different Full Disclosure reports.

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

Summary Statement

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

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

Numerical Quantities Summary

The numerical quantities listed below must be summarized near the beginning of the Full Disclosure Report.

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

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

Application Program

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

Appendix A contains the application source codes used in the TPC-C benchmark.

Sponsor

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

This benchmark test was sponsored by NEC Corporation . Packard Bell NEC has authorized NEC Corp. to publish TPC-C performance and price/performance results for the NEC Epress5800 HX4500. Price quotations contained in Appendix E correspond to the NEC Express5800 HX4500 server.

Parameters and Options

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

- *Database tuning options*
- *Recovery/locking options*
- *Operating system and application configuration parameters*

Appendix C contains the tunable parameters used in the TPC-C tests.

Configuration Diagrams

Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences. This includes, but not limited to:

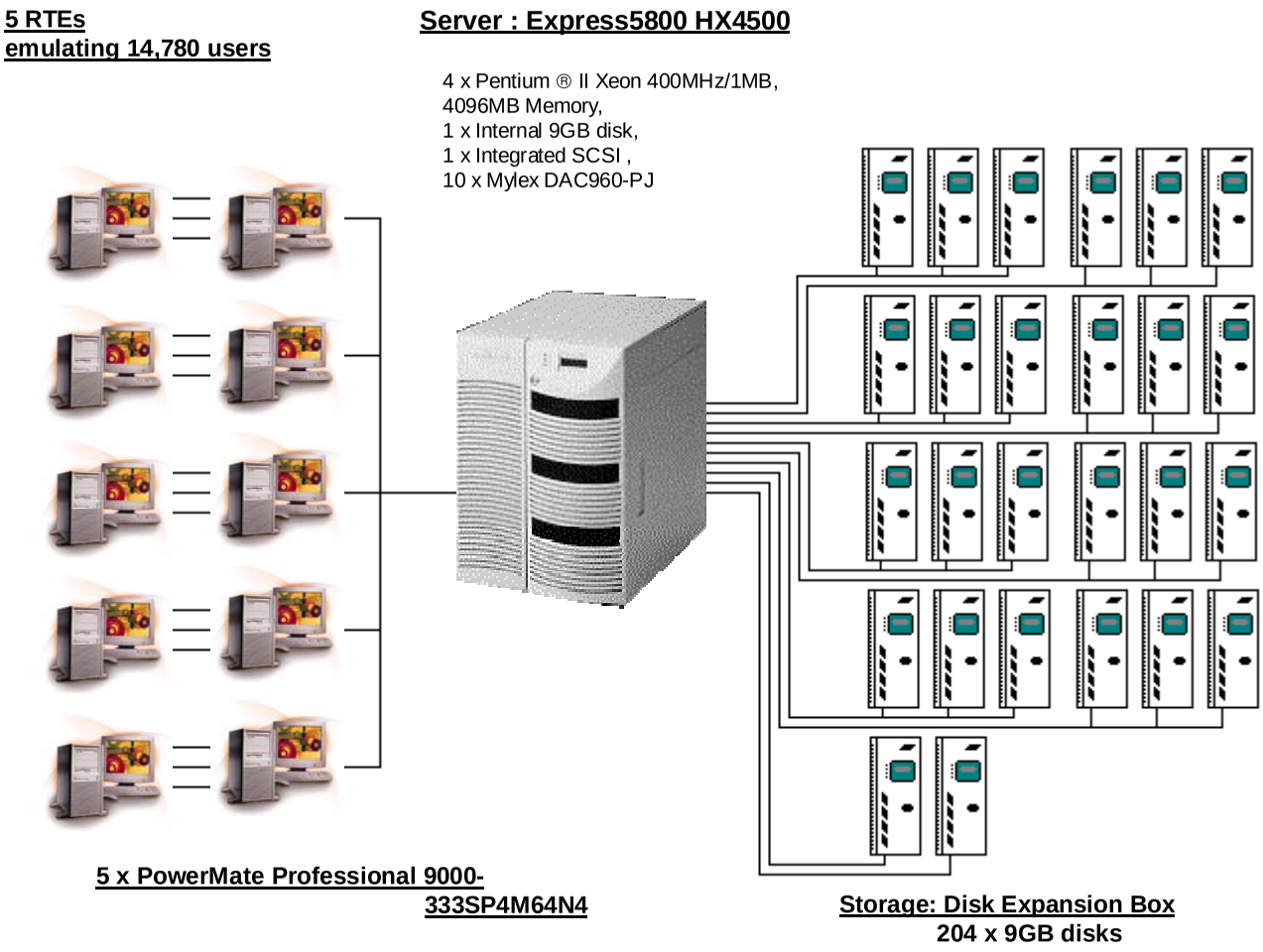
- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning or memory unique to the test*
- *Number and type of disk drive units (and controllers, if applicable)*
- *Number of channels or bus connections to disk units, including their protocol type*
- *Number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

Figure 1.1 shows the measured configuration diagram.
Figure 1.2 shows the priced configuration diagram.

Measured Configuration

The following figure represents the measured configuration. The benchmark system used a remote terminal emulator(RTE) to initiate transactions and measure response times of transactions, as well as record various data for each transaction.

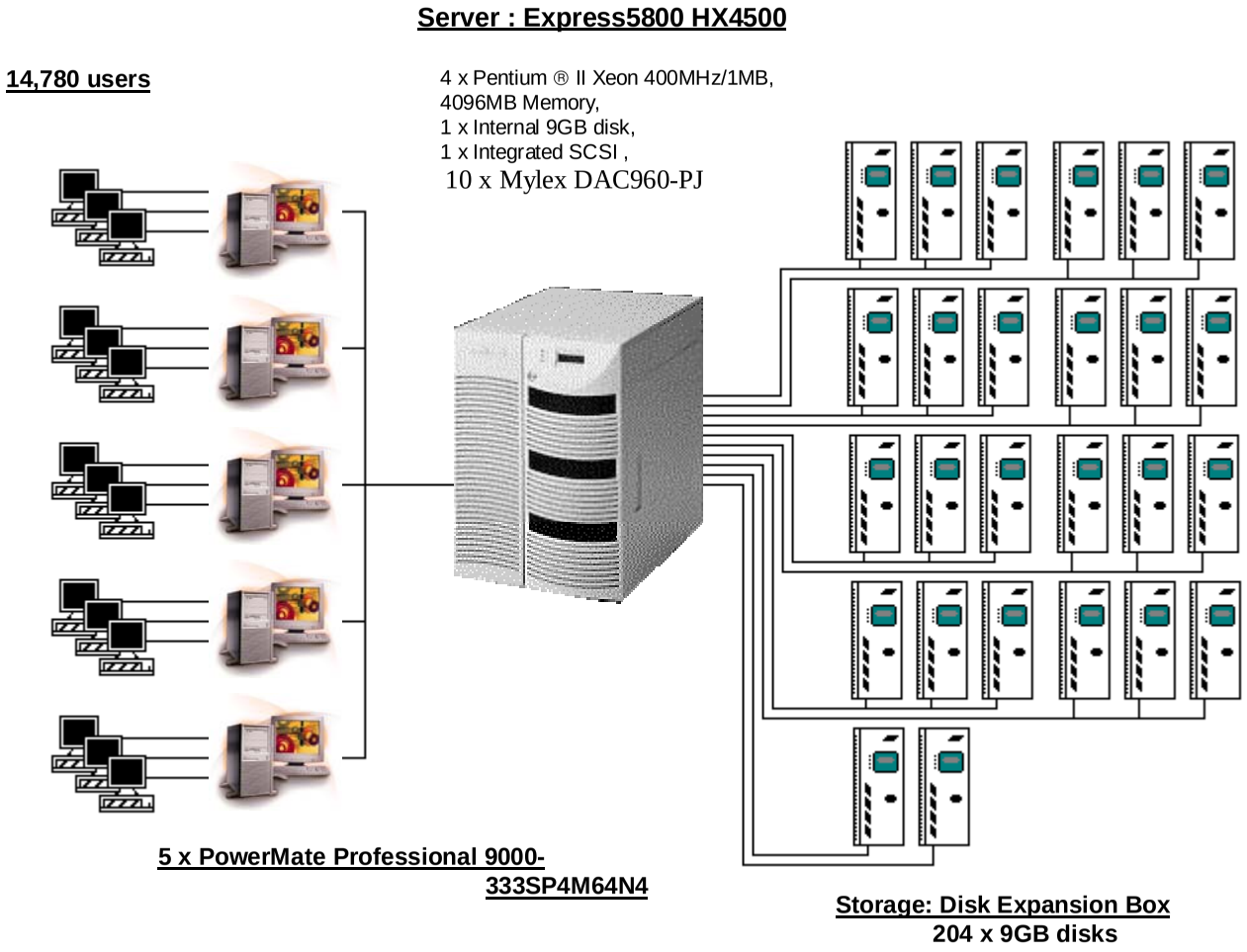
Figure 1.1 Express5800 HX4500, Measured Configuration Diagram



Priced System Configuration

The following figure depicts the priced system, whose cost determines the normalized price per tpmC reported for the test.

Figure1.2: Express5800 HX4500 , Priced Configuration Diagram



Clause 1 : Logical Database Design and Related Items

Table Definitions

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..

Table Organization

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

Appendix B contains the code used to define the physical organization of tables and indices

Insert and Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any 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 during the entire benchmark.

Disclosure of Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark(see Clause 1.6), any such partitioning must be disclosed.

Partitioning was not used on any table in this benchmark.

Replication of Tables

Replication of tables, if used, must be disclosed.

No tables were replicated in this benchmark test.

Additional and/or Duplicated Attributes in any Table

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

No duplications or additional attributes were used in this benchmark.

Clause 2 : Transaction and Terminal profiles Related Items

Random Number Generation

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

Random numbers were generated internaly by the Microsoft BenchCraft RTE program which was already audited independently.

Terminal Input/Output Screen Layout

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

All screen layouts followed the specifications exactly.

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 8.1.3.3.3 must be disclosed and commercially available (including supporting software and maintenance).

Each of five transaction types was tested by the auditor. The auditor verified that all the features specified in Clause 2.2.2.4 were provided.

Presentation Manager or Intelligent Terminal

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

Comment1: *The intent of this clause is to describe any special manipulations performed by a local terminal or workstation to off-load work from the SUT. This includes, but is not limited to : screen presentations, message bundling, and local storage of TPC-C rows.*
Comment2: *This disclosure also requires that all data manipulation functions also be described. Within this disclosure, the purpose of such additional function(s) must be explained.*

Application code running on the client machines implemented the TPC-C user interface. No presentation manager software or intelligent terminal features were used. The source code for the applications is listed in Appendix A.

Transaction Profiles

The percentage of home and remote order-lines in the New-Order transactions must be disclosed.
The percentage of New-Order transactions that were rolled back as a result of an unused item number must be disclosed.
The number of items per orders entered by New-Order transaction s must be disclosed.
The percentage of home and remote Payment transaction s must be disclosed.
The percentage of Payment and Order-Status transaction s that used non-primary key (C_LAST) access to the database must be disclosed.
The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW-ORDER table must be disclosed.

Table 1 shows the numerical quantities required by Clause 8.1.3.5 through 8.1.3.10.

Transaction Mix

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

Table 1 shows the mix of transaction types seen by the SUT during the reported measurement interval.

Following table summarizes the data required for disclosure in section 3.5 through 3.11.

Table 1 Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	**99.00%**
	Remote warehouse order lines	**1.00%**
	Rolled back transactions	**0.99%**
	Average items per order	**10.01**
Payment	Home warehouse payments	**84.96%**
	Remote warehouse payments	**15.04%**
	Accessed by last name	**60.09%**
Order Status	Accessed by last name	**60.17%**
Delivery	Skipped deliveries	**0**
Transaction Mix	New Order	**44.87%**
	Payment	**43.01%**
	Stock Level	**4.02%**
	Delivery	**4.06%**
	Order Status	**4.03%**

Queuing Mechanism

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

The client application processes submitted delivery transactions to named pipe delivery server software running on the client machines. There was a single delivery server with multiple execution threads running on each client machine. These delivery servers were responsible for processing deliveries queued to the named pipe and submitting them to the database server. The source code is listed in Appendix A.

Clause 3 : Transaction and System Properties Related Items

Transaction System Properties (ACID)

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

The TPC Benchmark™ C Standard Specification defines a set of transaction processing system properties that a system under test (SUT) must support during the execution of the benchmark. Those properties are Atomicity, Consistency, Isolation and Durability (ACID). This section quotes the specification definition of each of those properties and describes the tests done as specified and monitored by the auditor , to demonstrate compliance.

Atomicity Tests

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

Completed Transactions

Perform the Payment for randomly selected warehouse, district and customer(by customer number as specified in Clause 2.5.1.2) and verify that the records in the CUSTOMER,DISTRICT and WAREHOUSE tables have been changed appropriately.

The value of w_ytd, d_ytd, c_balance, c_ytd_payment and c_payment_cnt of a randomly selected warehouse, district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was committed. The values w_ytd, d_ytd, c_balance, c_ytd_payment, and c_payment_cnt were retrieved again. It was verified that all values had been changed appropriately.

Aborted Transactions

Perform the Payment transaction for randomly selected warehouse, district and customer(by customer number as specified in Clause 2.5.1.2) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that records in CUSTOMER, DISTRICT and WAREHOUSE tables have Not been changed.

The value of w_ytd, d_ytd, c_balance, c_ytd_payment and c_payment_cnt of randomly selected warehouse , district, and customer were retrieved. The Payment transaction was executed on the same warehouse, district, and customer. The transaction was rolled back. The values of w_ytd, d_ytd, c_balance, c_ytd_payment, c_payment_cnt were retrieved again. It was verified that none of the values had changed.

Consistency Tests

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

Consistency conditions one through four were tested using a script to issue queries to the database. The results of the queries verified that the database was consistent for all four tests. A run was executed over 10 minutes and included a checkpoint under 2000 users (200 active warehouse) condition . The shell script was executed before and after the run. The result of the same queries verified that the database remained consistent after the run.

Isolation

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation level is obtained.

Isolation tests one through nine were executed using shell scripts to issue queries to the database. Each script included timestamps to demonstrate the concurrency of operations. The results of the queries were captured to files. The captured files were verified to demonstrate the required isolation had been met.
Case A was followed for Isolation Test 7.

Durability

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

- *Permanent irrecoverable failure of any single durable medium containing database, ABTH files/tables, or recovery log data.*
- *Instantaneous interruption(system crash/system hang) in processing which requires system reboot to recover.*
- *Failure of all or part of memory(loss of contents)*

Loss of Memory and Loss of Log

Because the loss of power erases the contents of memory, both of instantaneous interruption and loss of memory were combined into a single test. Also loss of log was combined into the test.

The following steps were performed on a database of 1478 warehouses under the full load of users.

1. A sum of D_NEXT_O_ID of all rows in the district table was taken.
2. Full load of users were logged in to the database and running transactions for 5 minutes.
3. A checkpoint was initiated.
4. One disk drive of mirror paired drives holding the transaction log was removed.
   An NT alert message was displayed and logged in the NT event log.
   The benchmark run continued without interruption.
5. The running continued 2 minutes.
6. the system was powered off.
7. The RTE was shutted down.
8. The system was powered back up. SQL Server was restarted and automatically recovered.
9. A new count of D_NEXT_O_ID was taken.
10. This number was compared with the number of new orders reported by the RTE.

Loss of Data

Loss of data was demonstrated on a 10 Warehouse database for convenience. The standard driving mechanism was used to generate the transaction load of 100 users for the test. To demonstrate recovery from a permanent failure of durable media containing TPC-C tables, the following steps were performed. A fully scaled database would also pass this test.

1. A 10 Warehouse database was built having similar characteristics to the large database.
2. The database was backed up using SQL Server backup facilities.
3. A sum of D_NEXT_O_ID was taken.
4. 100 users were logged in to the database and running transactions.
5. One disk drive in the array was removed causing SQL Server error. SQL Server was shutted down.
6. SQL Server was restarted and a dump of the transaction log was taken.
7. The 10 Warehouse database was restored from backup.
8. The transaction log was restored and transactions rolled forward.
9. A new count of D_NEXT_O_ID was taken.
10. This number was compared with the number of new orders reported by the RTE.

Clause 4 : Scaling and Database Population Related Items

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.

The TPC-C database was originally built with 1520 warehouses. 42 warehouses were deleted prior to benchmark run.

Table 2 Number of Rows for Server

Table	Cardinality as benchmarked
Warehouse	1,478
Distinct	15,200
Customer	45,600,000
History	45,600,000
Orders	45,600,000
New Order	12,150,000
Order Line	456,003,360
Stock	152,000,000
Item	100,000
Deleted Warehouse Rows	42

Distribution of Tables and Logs

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

Table 3 depicts the distribution of the database over the disks of the tested system.

Figure 1.1, 1.2 shows the disk configuration for measured and priced system.

Table 3 : Data Distribution

Disk Administrator Configuration						
Disk	**Partition1**	**Partition2**	**Unused Space**	**HA#**	**LD#**	**Usage**
0 8676MB	D: NTFS 4087MB	C: NTFS 4581MB	(None)	1	1	System SQL Server
1 208392MB	G: (RAW) 10010MB	P: (RAW) 11100MB	187282MB	2	1	Data MSSQL70_misc1 MSSQL70_cstock8
2 208392MB	H: (RAW) 10010MB	O: (RAW) 11100MB	187282MB	3	1	Data MSSQL70_misc2 MSSQL70_cstock7
3 208392MB	S: (RAW) 10010MB	N: (RAW) 11100MB	187282MB	4	1	Data MSSQL70_misc3 MSSQL70_cstock6
4 52098MB	F: (RAW) 52098MB		(None)	5	1	Log (Mirrored)
5 121492MB	W: NTFS 121492MB		(None)	6	1	DB backup
6 208392MB	U: (RAW) 10010MB	L: (RAW) 11100MB	187282MB	7	1	Data MSSQL70_misc5 MSSQL70_cstock4
7 208392MB	T: (RAW) 10010MB	M: (RAW) 11100MB	187282MB	8	1	Data MSSQL70_misc4 MSSQL70_cstock5
8 52098MB	F: (RAW) 52098MB		100647MB	9	1	Log (Mirrored)
9 208392MB	Q: (RAW) 10010MB	J: (RAW) 11100MB	187282MB	10	1	Data MSSQL70_misc7 MSSQL70_cstock2
10 208392MB	R: (RAW) 10010MB	I: (RAW) 11100MB	187282MB	11	1	Data MSSQL70_misc8 MSSQL70_cstock1
11 208392MB	V: (RAW) 10010MB	K: (RAW) 11100MB	187282MB	12	1	Data MSSQL70_misc6 MSSQL70_cstock3

Type of Database

A statement must be provided that describes:
1)The data model implemented by DBMS used (e.g. relational, network, hierarchical).
2)The database interface (e.g. embedded, call level) and access language (e.g. SQL, PL/1, COBOL read/write used to implement the TPC-C transaction. If more than one interface/access language is used to implement TPC-C, each interface/access language must be described and a list of which interface/access language is used with which transaction type must be disclosed.

Microsoft SQL Server, Enterprise Edition 7.0, a relational database, was used in this benchmark. SQL Server stored procedures were used and invoked through DB-Library function calls embedded in C code.

Database Mapping

The mapping of database partitions/replications must be explicitly described.
No partitioning or replication was used.

180-Days Space

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

The detail of 180-day space calculation is shown in Appendix D.
To calculate the space required to sustain the database log for 8 hours of growth at steady state, the following steps were followed:

1. The free space on the log file was queried using *DBCC sqlperf(logspace).*
2. Transactions were run against the database with a full load of users.
3. The free space was again queried using *DBCC sqlperf(logspace).*
4. The space used was calculated as the difference between the first and second query.
5. The number of NEW-ORDERS was verified from an RTE report covering the entire run.
6. The space used was divided by the number of NEW-ORDERS giving a spaceused per NEW-ORDER transaction.
7. The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

The results of the above steps yielded a requirement of 45.47 GB to sustain the log for 8 hours.
Space available on the transaction log volume was 101.68 GB (including mirror), indicating that enough storage was configured to sustain 8 hours of growth.

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

Clause 5 : Performance Metrics and Response Time Related Items

Throughput

Measured tpmC must be reported

Table 4 : Measured tpmC

18322.67 tpmC

Response Times

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

Table 5 : Response Times (in seconds)

Type	Average	Maximum	90th %
New-Order	0.46	17.01	0.70
Payment	0.32	9.35	0.56
Stock Level	2.40	8.28	2.92
Interactive Delivery	0.29	0.57	0.30
Deferred Delivery	0.58	17.63	0.89
Order Status	0.40	11.78	0.63
Menu	0.20	1.02	0.31

Keying and Think Times

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

Table 6 : Keying Times

Type	Minimum	Average	Maximum
New-Order	18.00	18.01	18.03
Payment	3.00	3.01	3.03
Stock Level	2.00	2.01	2.03
Interactive Delivery	2.00	2.01	2.02
Order Status	2.00	2.01	2.03

Table 7 : Think Times

Type	Minimum	Average	Maximum
New-Order	0.00	12.05	120.71
Payment	0.00	12.08	120.71
Stock Level	0.00	5.07	48.46
Interactive Delivery	0.00	5.08	50.70
Order Status	0.00	10.08	100.70

Response Time Frequency Distribution Curves and Other Graphs

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

Figure 2.1 : New Order Response Time Distribution

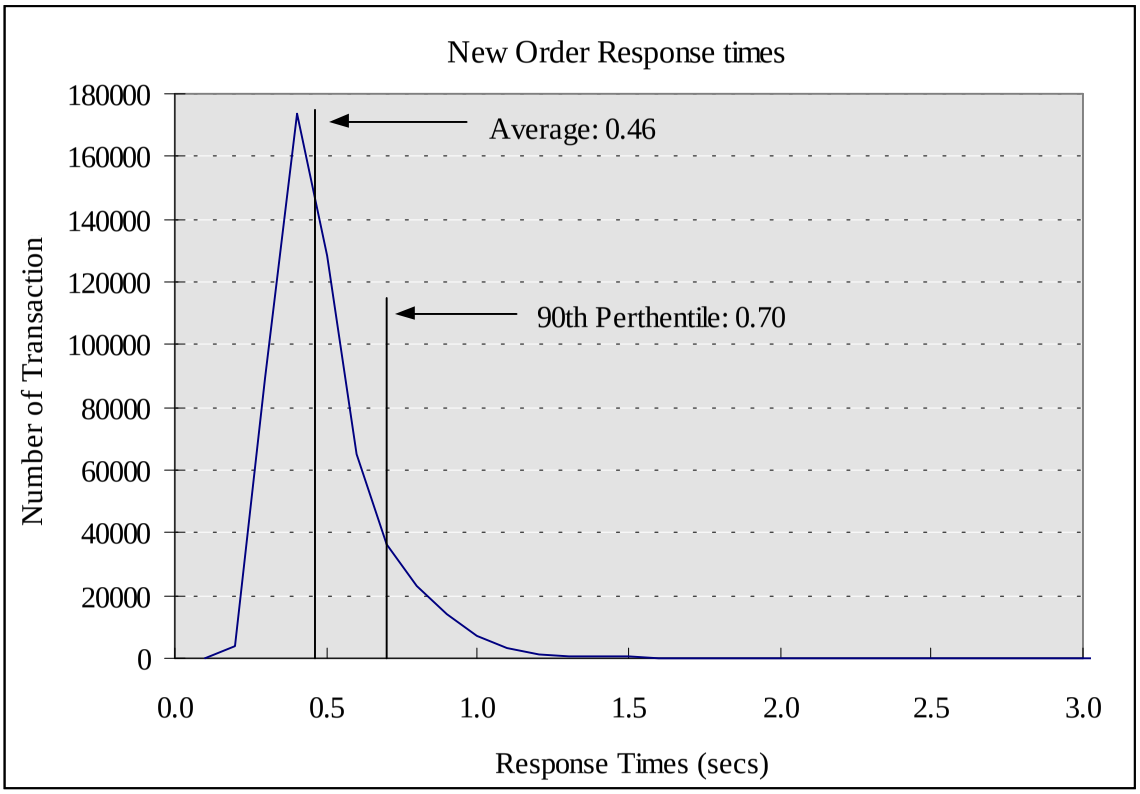


Figure 2.2 : Payment Response Time Distribution

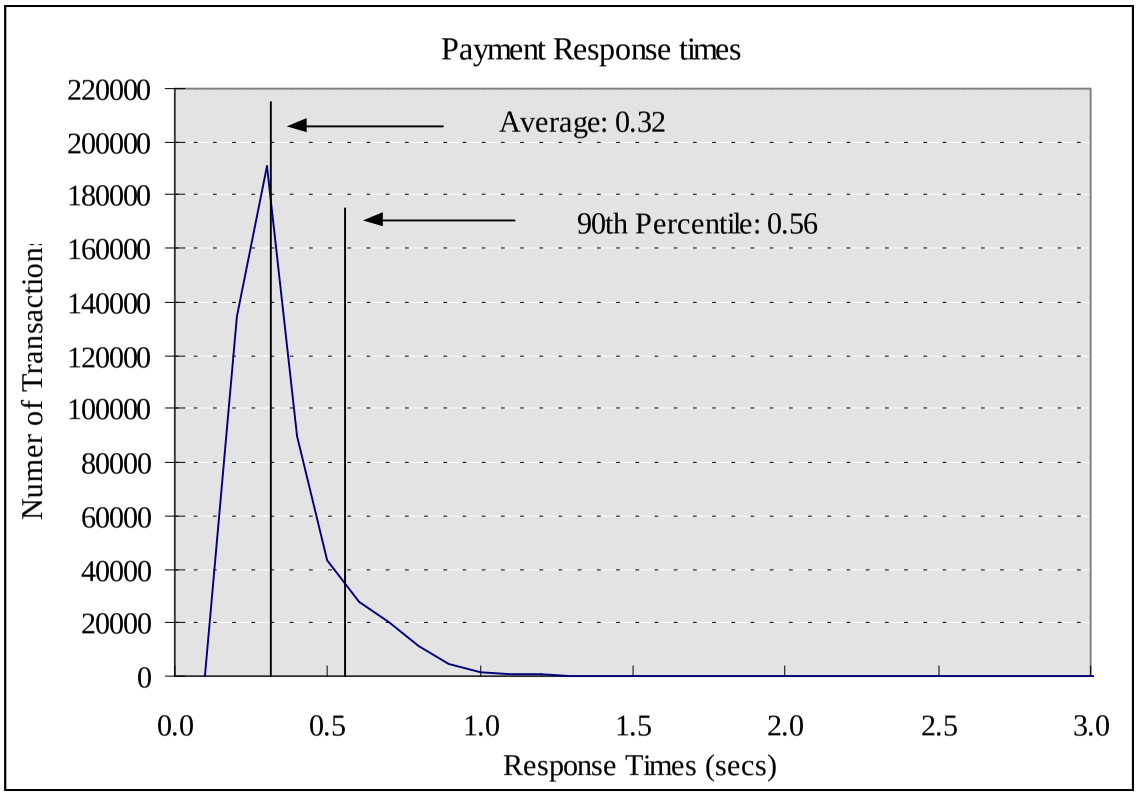


Figure 2.3 : Order Status Response Time Distribution

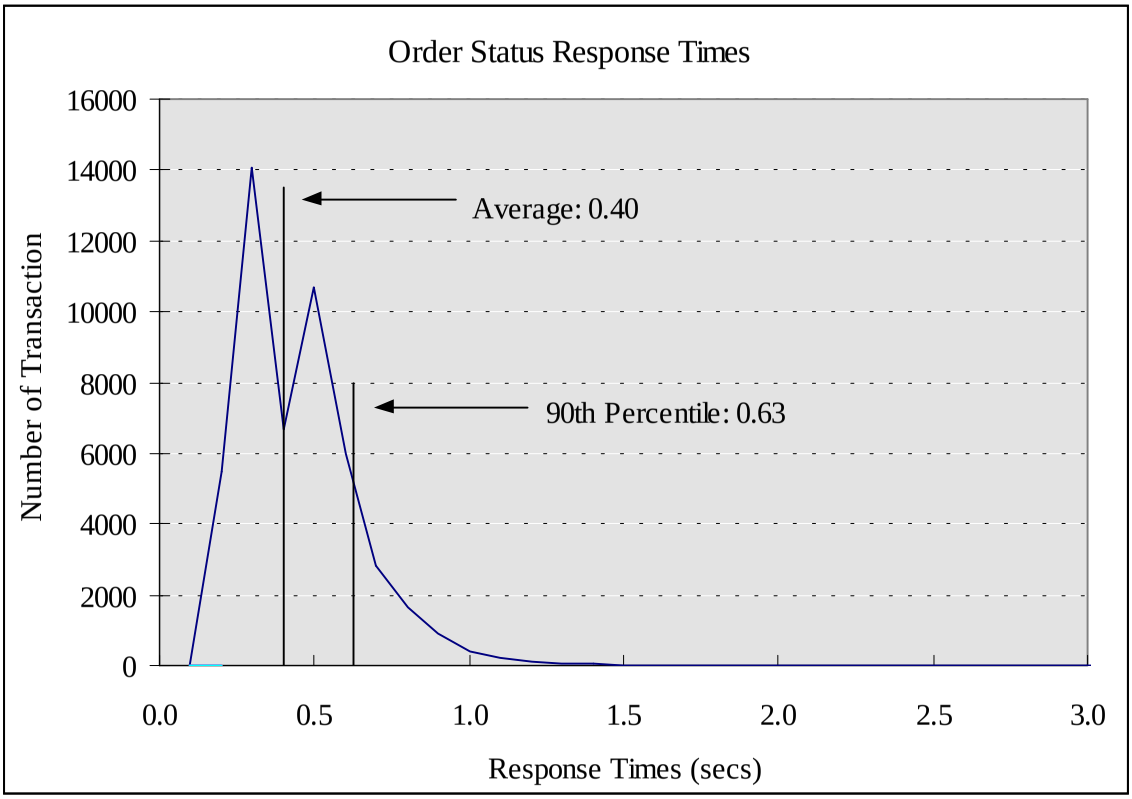


Figure 2.4 : Delivery Response Time Distribution

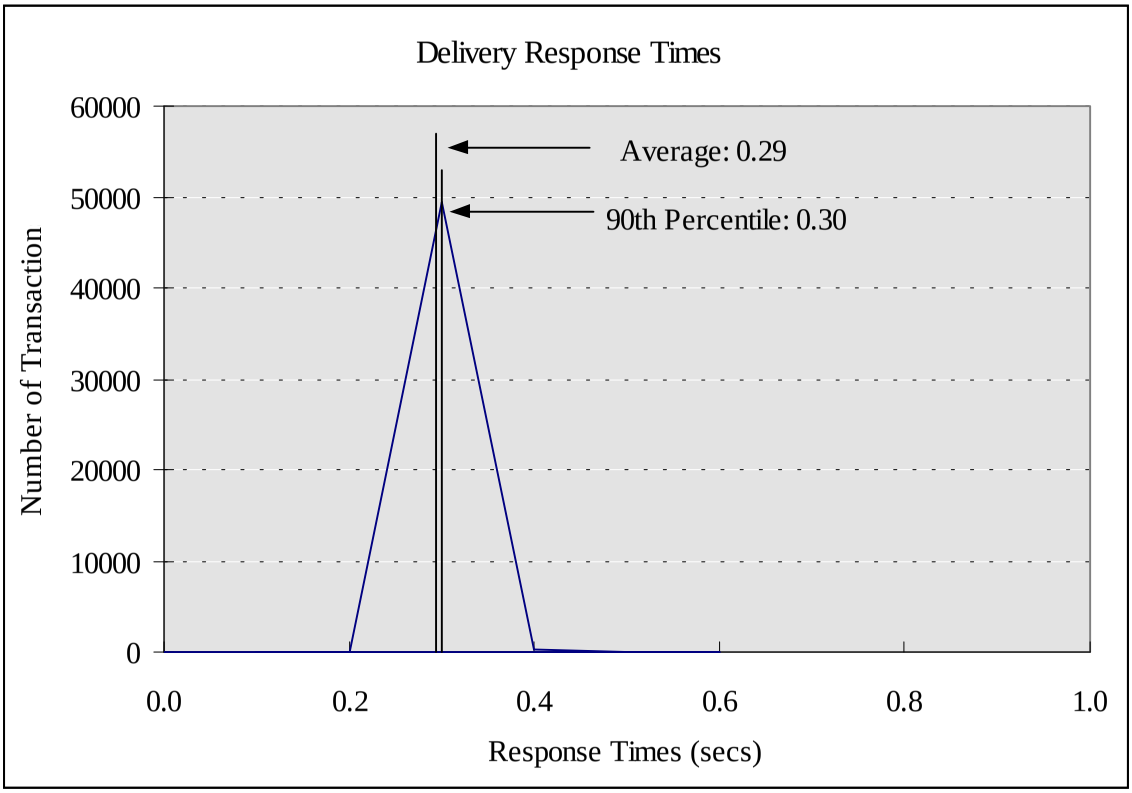
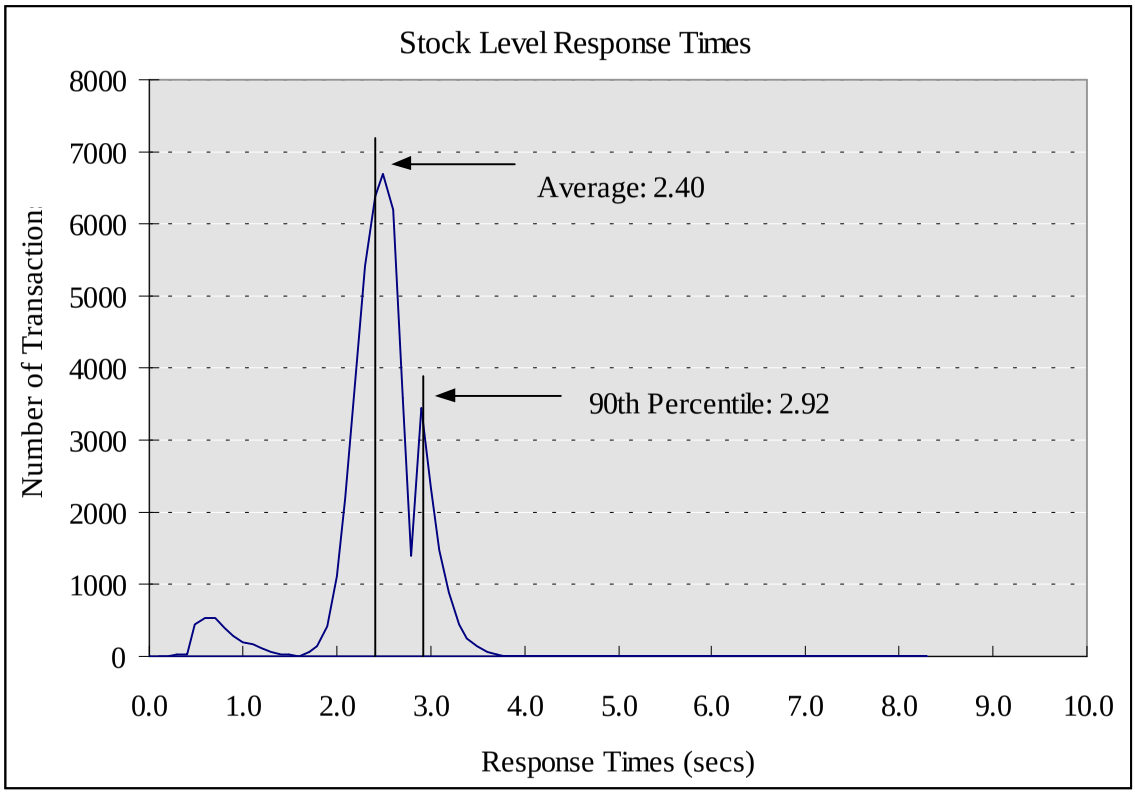


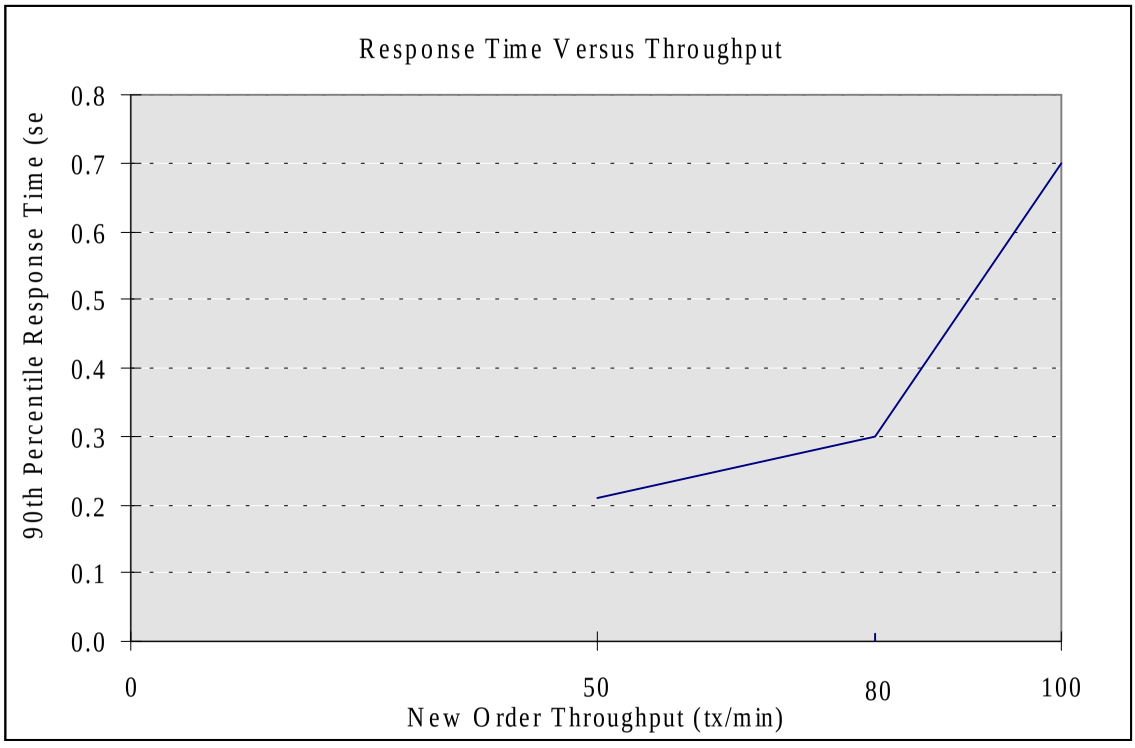
Figure 2.5 : Stock Level Response Time Distribution



Response time versus Throughput Performance Curve

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

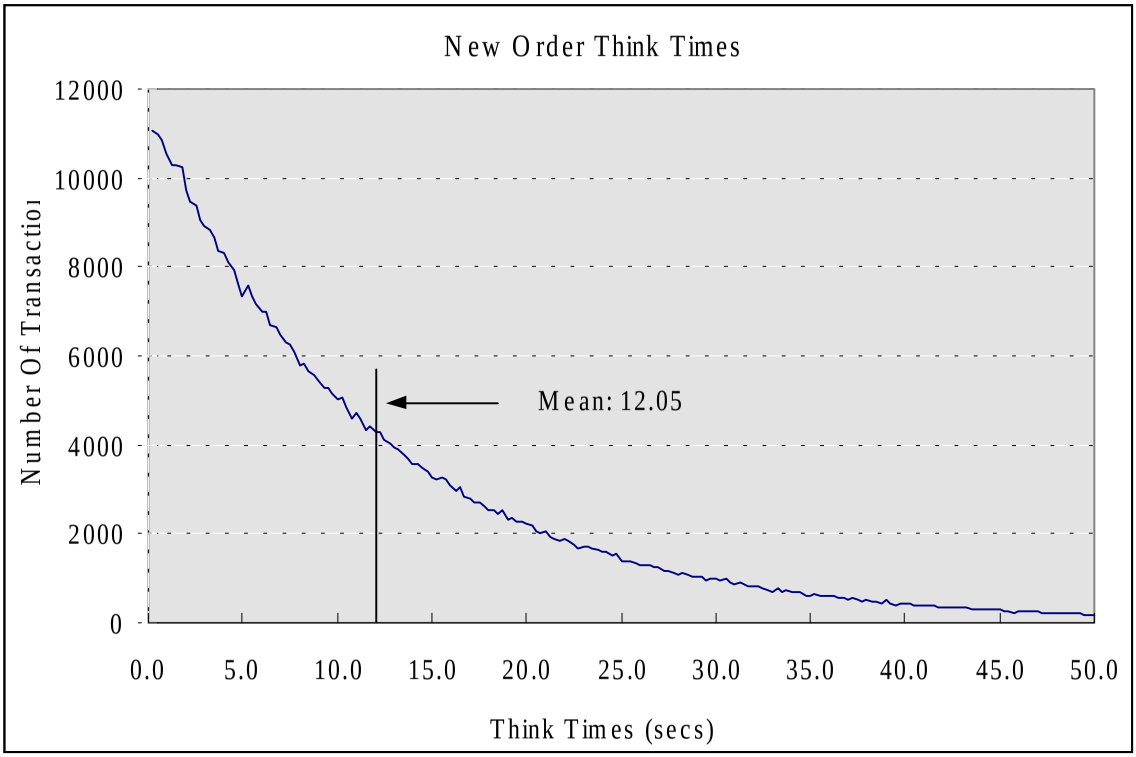
Figure 2.6 Response Time Performance vs. Throughput Curve



NEW-Order Think Time

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

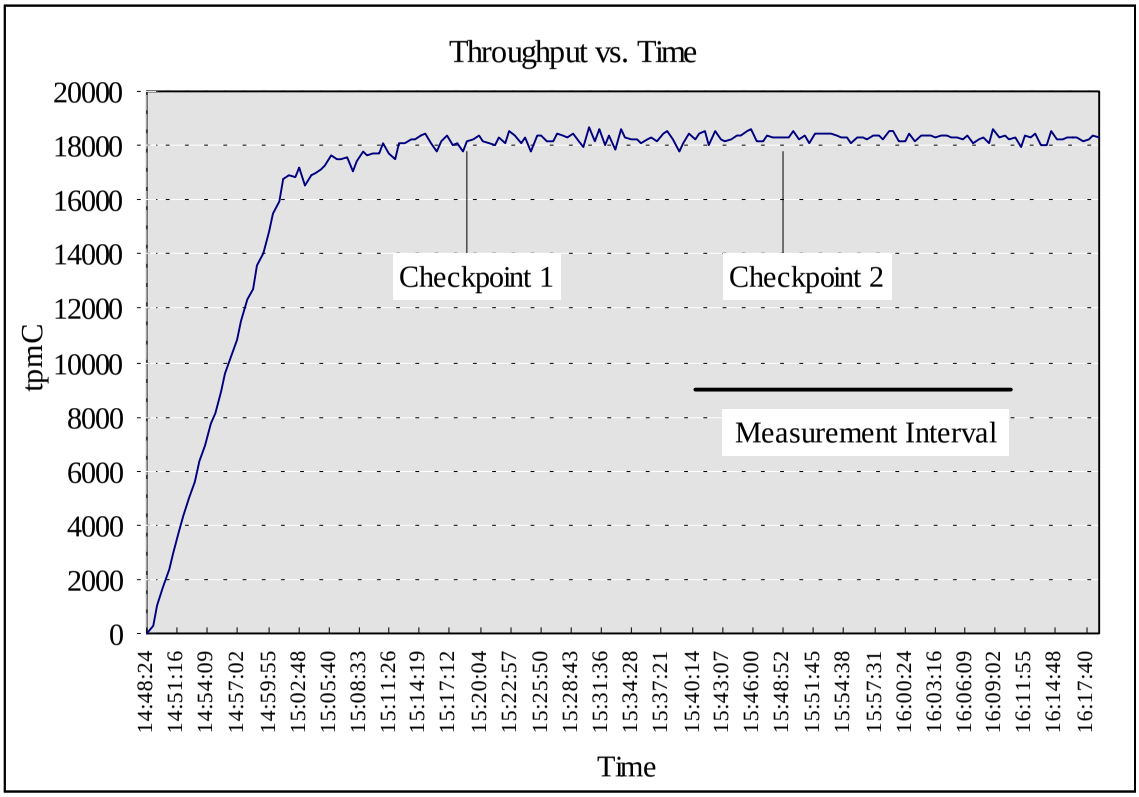
Figure 2.7 New-Order Think Time



New-Order Throughput vs. Elapsed Time

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

Figure 2.8 New Order Throughput vs. Time



Steady State

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

Steady state was confirmed by the throughput data collected during the run and graphed in Figure 2.8.

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.

A checkpoint in Microsoft SQL Server writes to disk all updated memory pages that have not been yet actually written to disk. SQL Server recovery interval parameter was set to the maximum allowable value to perform checkpoint at specific intervals. A checkpoint script, which issues specified number of checkpoint at specified (30 minutes) intervals, was started after all users logged in and sending transactions.

Reproducibility

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

The reproducibility test result is taken from another, non-overlapping, measurement interval of the same duration as the reported interval. The throughput difference measured over that interval was within 0.17% of reported interval result.

Measurement Period Duration

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

The reported measured interval was exactly 30 minutes long.

Regulation of Transaction Mix

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

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

Transaction Statistics

The percentage of the total mix for each transaction type must be disclosed. The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed. The average number of order-lines entered per New-Order transaction must be disclosed. The percentage of remote order lines per New-Order transaction must be disclosed. The percentage of remote Payment transactions must be disclosed. The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

The above statistics are disclosed in Table 1.

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.

Initial checkpoint was started 30 minutes after the start of ramp-up. Second checkpoint was started 30 minutes after the 1st checkpoint. The time from the start of the Measurement interval was 21.7 minutes after. In accord with Clause 5.5.22, there is no checkpoint within the “guard zones” 1800/4=450 seconds from the beginning and end of the measurement interval.

Clause 6 : SUT, Driver, and Communication Definition Related Items

Descriptions of RTE

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

The RTE used was the Microsoft BenchCraft RTE System. The RTE input parameters are listed in Appendix C.

Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to the priced system. The results of the test described in Clause 6.6.3.4 must be disclosed..

AS configured for this test, the driver software emulates the traffic that would be observed from the users' PCs connected by Ethernet to the front-end clients using HTTP (HyperText Transfer Protocol) over TCP/IP. One tenth of a second (100 milli seconds) was added to each transaction time to compensate for the overhead of the Web browser.

Functional Diagrams and Detail of Driver System

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

The diagrams in figure 1.1 and 1.2 show the tested and priced benchmark configurations.

Network configurations and Driver system

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

Figure 1.1 and 1.2 in this report has the network configurations of both the tested system and the priced system.

The front-end clients were connected over one 100Mbps 100Base-T Ethernet segments to the back-end. Each front-end client were connected to the RTE over three 10Mbps 10Base-T Ethernet segments.

The priced PCs are also connected using 10Mbps Ethernet to the front-end clients.

Network Bandwidth

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

The Ethernet used in the local area network (LAN) between the emulated terminals and the front-end system compiles with the IEEE 802.3 standard and has a bandwidth of 10Mbps.

Operator Intervention

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

This configuration does not require any operator intervention to sustain eight hours of the reported throughput.

Clause 7 : Pricing Related Items

Hardware and Software Components

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, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

The detailed list of all hardware and software for the priced configuration is listed in the system pricing summary.

Availability

The committed delivery date for general availability (availability date) of products used in the price calculation must be reported. When the priced system included products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available. The single date must be reported on the first page of the Executive Summary. All availability dates,whether for individual components or for the SUT as a whole, must be disclosed to a precision of one day.

All the components used in the priced system are currently available with the exception of:

NEC Express5800 HX4500 will be available by October 2, 1998.
Microsoft SQL Server Enterprise Edition7.0 will be available by December 29, 1998.

Throughput, and Price Performance

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

- Maximum Qualified Throughput 18322.67 tpmC
- Price per tpmC : $33.55 per tpmC
- Total 5-yesr cost of ownership

Country Specific Pricing

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

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

Usage Pricing

For any usage pricing, the sponsor must disclose:

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

None

Clause 9 : Audit Related Items

Auditor's Report

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

Next page contains the complete independent auditor's report by Francois Raab of Information Paradigm Inc. for the test described in this report.

Availability of the Full Disclosure Report

The Full Disclosure Report must be readily available to the public at a reasonable charge, similar to the charges for similar documents by the test sponsor. The report must be made available when results are made public. In order to use the phrase "TPC Benchmark™ C", the Full Disclosure Report must have been submitted to the TPC Administrator as well as written permission obtained to distribute same.

Requests for this TPC Benchmark™ C Full Disclosure Report should be sent to:
Transaction Processing Performance Council
c/o Shanley Public Relations
777 North First Street, Suite 6000
San Jose, CA 95112-6311
or your local NEC / Packard Bell -NEC office.

Auditor's letter

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Test Sponsor: Eiichi Kenai
NEC Corporation
3rd Development Dept
3rd Computers Software Dept
Fuchu City Tokyo 183, Japan

August 3, 1998

I verified the TPC Benchmark[a] C performance of the following configuration:

Platform: Express 5800 HX4500 c/s
DataBase Manager: Microsoft SQL Server 7.0 Enterprise Edition
Operating System: Microsoft Windows NT 4.0 Enterprise Edition
Transaction Monitor: BEA Tuxedo Version 6.3CFS
Other Software: Microsoft IIS 3.0

The results were:

CPU's	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: Express 5800 HX4500				
4 x Intel Pentium II Xeon (400 MHz)	1 MB L2 4096 MB Main	204 x 9 GB ext. 1 x 9 GB int.	0.70 Seconds	18,322.67
Five Clients: PowerMate Enterprise 900-333SP (Specification for each)				
2 x Intel Pentium II (333 MHz)	512 KB L2 512 MB Main	1 x 4.5 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for Revision 3.3.2 of the benchmark. The following verification items were given special attention:

- The transactions were correctly implemented
- The database records were the proper size
- The database was properly scaled and populated
- The ACID properties were met

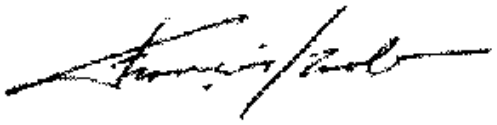
1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office : 719/473-7555 • Fax : 719/473-7554

- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 30 minutes
- One checkpoint was taken during the measurement interval
- Measurement repeatability was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,



François Raab
President

1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office : 719/473-7555 • Fax : 719/473-7554

Appendix A : Application Source Code

Makefile

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

!IF "$(SQL_LOC)" == ""
SQL_LOC=C:\MSSQL\DBLIB
!MESSAGE No SQL_LOC specified.  Defaulting to C:\MSSQL\DBLIB
!ENDIF

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

OUTDIR          = .
SRCDIR          = .\Src
OBJDIR          = .\Objs
OUTDIR          = .\Bin
ODBC                            = \odbcsdk

DBLIB           = $(SQL_LOC)
DBLIBINC        = $(DBLIB)\INCLUDE
ODBCINCDIR                      = $(ODBC)\INCLUDE
DBLIBDIR        = $(DBLIB)\LIB
ODBCLIBDIR                      = $(ODBC)\LIB32

!IF "$(CFG)" != "Debug"
LDEBUG      =
CDEBUG      =
LDEBUG_RG    =
CDEBUG_RG    =
DEBUG       =
FLAGS       = /D "WIN32" /D "_WINDOWS"
OPT         = /Ot
!ELSE
LDEBUG      = /debug /pdb:$(OBJDIR)\tpcc1.pdb
CDEBUG      = /Zi /Yd
LDEBUG_RG    = /debug /pdb:$(OBJDIR)\install.pdb
CDEBUG_RG    = /Zi /Yd /Fd$(OBJDIR)\install.pdb
FLAGS       = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
OPT         = /Od
!ENDIF

LINK32_LIBS1    = user32.lib msacm32.lib advapi32.lib
LINK32_OBJS1    = "$(OBJDIR)\tpcc1.obj" "$(OBJDIR)\tpcc1.res"
"$(OBJDIR)\tux_sql.obj" "$(OBJDIR)\error.obj" "$(OBJDIR)\util.obj"
"$(OBJDIR)\pipe_routines.obj"
LINK32_DEF1     = "$(SRCDIR)\tpcc1.def"
LINK32_FLAGS1   = /nologo /subsystem:windows /dll /incremental:no
$(LDEBUG) /def:"$(LINK32_DEF1)" /out:"$(OBJDIR)\tpcc1.dll"

LINK32_LIBS2    = user32.lib msacm32.lib advapi32.lib
$(ODBCLIBDIR)\odbc32.LIB
LINK32_OBJS2    = "$(OBJDIR)\tpcc2.obj" "$(OBJDIR)\tpcc2.res"
LINK32_DEF2     = "$(SRCDIR)\tpcc2.def"
LINK32_FLAGS2   = /nologo /subsystem:windows /dll /incremental:no
$(LDEBUG) /def:"$(LINK32_DEF2)" /out:"$(OBJDIR)\tpcc2.dll"

LINK32_LIBS_RG  = user32.lib gdi32.lib advapi32.lib version.lib comctl32.lib
LINK32_OBJS_RG  = "$(OBJDIR)\install.obj" "$(OBJDIR)\install.res"
LINK32_FLAGS_RG = /nologo /subsystem:windows /incremental:no
$(LDEBUG_RG) /out:$(OUTDIR)\install.exe

ALL:    $(OBJDIR)\. $(OUTDIR)\. $(OUTDIR)\install.exe

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

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

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

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

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

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

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

$(OBJDIR)\tpcc1.res:    $(SRCDIR)\tpcc1.rc
                rc.exe /l 0x409 /fo $(OBJDIR)\tpcc1.res $(FLAGS)
$(SRCDIR)\tpcc1.rc

$(OBJDIR)\tpcc1.dll:    $(LINK32_OBJS1) $(LINK32_DEF1)
                link.exe  $(LINK32_FLAGS1) $(LINK32_OBJS1) $(LINK32_LIBS1)

"$(OBJDIR)\tpcc2.obj":  "$(SRCDIR)\tpcc.c" "$(SRCDIR)\tpcc.h"
                cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(ODBCINCDIR)
$(FLAGS) /Fd$(OBJDIR)\tpcc2.pdb /Fo$(OBJDIR)\tpcc2.obj /c /D"USE_ODBC"
"$(SRCDIR)\tpcc.c"

$(OBJDIR)\tpcc2.res:    $(SRCDIR)\tpcc2.rc
                rc.exe /l 0x409 /fo $(OBJDIR)\tpcc2.res $(FLAGS)
$(SRCDIR)\tpcc2.rc

$(OBJDIR)\tpcc2.dll:    $(LINK32_OBJS2) $(LINK32_DEF2)
                link.exe  $(LINK32_FLAGS2) $(LINK32_OBJS2) $(LINK32_LIBS2)

$(OBJDIR)\delisrv1.exe:  $(SRCDIR)\delisrv.c $(SRCDIR)\delisrv.h
                cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I $(DBLIBINC)
$(FLAGS) /Fo$(OBJDIR)\delisrv.obj $(SRCDIR)\delisrv.c /link
/out:$(OBJDIR)\delisrv1.exe $(DBLIBDIR)\ntwdblib.lib msacm32.lib advapi32.lib

$(OBJDIR)\delisrv2.exe:  $(SRCDIR)\delisrv.c $(SRCDIR)\delisrv.h
                cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I$(ODBCINCDIR)
$(FLAGS) /Fo$(OBJDIR)\delisrv.obj $(SRCDIR)\delisrv.c /D"USE_ODBC"  /link
/out:$(OBJDIR)\delisrv2.exe $(ODBCLIBDIR)\odbc32.lib msacm32.lib
advapi32.lib

$(OBJDIR)\install.res:  $(SRCDIR)\install.rc $(OBJDIR)\tpcc1.dll
$(OBJDIR)\delisrv1.exe
                rc.exe /l 0x409 /fo$(OBJDIR)\install.res /i $(OBJDIR) /i $(SRCDIR)
$(FLAGS) $(SRCDIR)\install.rc

$(OBJDIR)\install.obj:  $(SRCDIR)\install.c $(OBJDIR)\tpcc1.dll
$(OBJDIR)\delisrv1.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)
<<
```

cl_build.bat

```
set CFLAGS=/MT /O2
buildclient -o tux_client -f tux_client.c -f util.c -f getopt.c -f pipe_routines.c -v
```

Sv_build.bat

```
set CFLAGS=/MD /O2
buildserver -o tux_server  -f \mssql\dblib\lib\ntwdblib.lib -f tux_server.c -f
sql_routines.c -f error.c -f util.c -s NEW_ORDER -s PAYMENT -s
ORDER_STATUS -s STOCK_LEVEL
```

tpcc.def

```
LIBRARY TPCC.DLL

EXPORTS

                GetExtensionVersion     @1
                HttpExtensionProc       @2
```

tpcc1.def

```
LIBRARY TPCC1.DLL

EXPORTS

                GetExtensionVersion     @1
                HttpExtensionProc       @2
```

tpcc2.def

```
LIBRARY TPCC2.DLL
```

```
EXPORTS

        GetExtensionVersion     @1
        HttpExtensionProc       @2
```

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
```

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
        blnUse;                         //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
```

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
```

getopt.h

```
#ifndef _GETOPT_H_INCLUDED
#define _GETOPT_H_INCLUDED
#endif
```

Httpext.h

```
/********
*
*  Copyright (c) 1995  Process Software Corporation
*
*  Copyright (c) 1995  Microsoft Corporation
*
*
*  Module Name  : HttpExt.h
*
*  Abstract :
*
*     This module contains  the structure definitions and prototypes for the
*     version 1.0 HTTP Server Extension interface.
*
*****************/

#ifndef _HTTPEXT_H_
#define _HTTPEXT_H_

#include <windows.h>

#ifdef __cplusplus
extern "C" {
#endif

#define   HSE_VERSION_MAJOR          1     // major version of this spec
#define   HSE_VERSION_MINOR          0     // minor version of this spec
#define   HSE_LOG_BUFFER_LEN         80
#define   HSE_MAX_EXT_DLL_NAME_LEN  256

typedef   LPVOID  HCONN;

// the following are the status codes returned by the Extension DLL

#define   HSE_STATUS_SUCCESS                    1
#define   HSE_STATUS_SUCCESS_AND_KEEP_CONN      2
```

```
#define   HSE_STATUS_PENDING                3
#define   HSE_STATUS_ERROR                  4

// The following are the values to request services with the
ServerSupportFunction.
//  Values from 0 to 1000 are reserved for future versions of the interface

#define   HSE_REQ_BASE                      0
#define   HSE_REQ_SEND_URL_REDIRECT_RESP    ( HSE_REQ_BASE
+ 1 )
#define   HSE_REQ_SEND_URL                  ( HSE_REQ_BASE + 2 )
#define   HSE_REQ_SEND_RESPONSE_HEADER      ( HSE_REQ_BASE
+ 3 )
#define   HSE_REQ_DONE_WITH_SESSION         ( HSE_REQ_BASE + 4 )
#define   HSE_REQ_END_RESERVED              1000

//
//  These are Microsoft specific extensions
//

#define   HSE_REQ_MAP_URL_TO_PATH
(HSE_REQ_END_RESERVED+1)
#define   HSE_REQ_GET_SSPI_INFO
(HSE_REQ_END_RESERVED+2)


//
// passed to GetExtensionVersion
//

typedef struct   _HSE_VERSION_INFO {

    DWORD  dwExtensionVersion;
    CHAR   lpszExtensionDesc[HSE_MAX_EXT_DLL_NAME_LEN];

} HSE_VERSION_INFO, *LPHSE_VERSION_INFO;

//
// passed to extension procedure on a new request
//
typedef struct _EXTENSION_CONTROL_BLOCK {

    DWORD     cbSize;               // size of this struct.
    DWORD     dwVersion;            // version info of this spec
    HCONN     ConnID;               // Context number not to be modified!
    DWORD     dwHttpStatusCode;     // HTTP Status code
    CHAR      lpszLogData[HSE_LOG_BUFFER_LEN];// null terminated log info
specific to this Extension DLL

    LPSTR     lpszMethod;           // REQUEST_METHOD
    LPSTR     lpszQueryString;      // QUERY_STRING
    LPSTR     lpszPathInfo;         // PATH_INFO
    LPSTR     lpszPathTranslated;   // PATH_TRANSLATED

    DWORD     cbTotalBytes;         // Total bytes indicated from client
    DWORD     cbAvailable;          // Available number of bytes
    LPBYTE    lpbData;              // pointer to cbAvailable bytes

    LPSTR     lpszContentType;      // Content type of client data

    BOOL (WINAPI *GetServerVariable) ( HCONN      hConn,
                                       LPSTR      lpszVariableName,



                                                                  LPVOID
lpvBuffer,
                                       LPDWORD    lpdwSize );

    BOOL (WINAPI *WriteClient)  ( HCONN      ConnID,
                                  LPVOID     Buffer,
                                  LPDWORD    lpdwBytes,
                                  DWORD      dwReserved );

    BOOL (WINAPI *ReadClient)  ( HCONN      ConnID,
                                 LPVOID     lpvBuffer,
                                 LPDWORD    lpdwSize );

    BOOL (WINAPI *ServerSupportFunction)( HCONN      hConn,
                                          DWORD      dwHSERRequest,
                                          LPVOID     lpvBuffer,
                                          LPDWORD    lpdwSize,
                                          LPDWORD    lpdwDataType );

} EXTENSION_CONTROL_BLOCK, *LPEXTENSION_CONTROL_BLOCK;

//
//  these are the prototypes that must be exported from the extension DLL
//

BOOL  WINAPI   GetExtensionVersion( HSE_VERSION_INFO  *pVer );
DWORD WINAPI   HttpExtensionProc(  EXTENSION_CONTROL_BLOCK
*pECB );

// the following type declarations is for the server side

typedef BOOL  (WINAPI *PFN_GETEXTENSIONVERSION)(
HSE_VERSION_INFO  *pVer );
typedef DWORD (WINAPI *PFN_HTTPEXTENSIONPROC )(
EXTENSION_CONTROL_BLOCK *pECB );

#ifdef __cplusplus
}
#endif

#endif  // end definition _HTTPEXT_H_
```

install.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
```

Pipe_routines.h

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

resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by TPCC.rc
//

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        101
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

sqlroutines.h

```
// dblib error message return values //--@add
#define INT_EXIT 0
#define INT_CONTINUE 1
#define INT_CANCEL 2

// 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
        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);
int SQLStockLevel( EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry);
int     SQLNewOrder( EXTENSION_CONTROL_BLOCK* pECB, int
iTermId, int iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA *pNewOrder,
short deadlock_retry);
int SQLPayment( EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
deadlock_retry);
int SQLOrderStatus( EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry);
BOOL SQLInit( void);
void SQLCleanup( void);
BOOL SQLThreadAttach( void);
BOOL SQLThreadDetach( void);
PECBINFO SQLGetECB( PDBPROCESS p);
```

Tpcc.h

```
#ifndef TPCC_H_INCLUDED
        #define TPCC_H_INCLUDED
        extern char szErrorLogPath[];
        #ifdef TUX    //--@del
//      #ifndef TUX   //--@add
                #include "tpcc_tux.h"
        #else
                #include <httpext.h>
                #include "tpcc_real.h"
        #endif
#endif
```

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 blnput, BOOL
bValid);
static char *MakePaymentForm( int iTermId, int iSyncId, BOOL blnput);
static char *MakeOrderStatusForm( int iTermId, int iSyncId, BOOL blnput);
static char *MakeDeliveryForm( int iTermId, int iSyncId, BOOL blnput, 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;

//BOOL IsValidTermId(int TermId);
```

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
```

trans.h

```
/* FILE: TRANS.H
* Microsoft TPC- C Kit Ver.3.00.000
* Audited 08/ 23/ 96By Francois Raab
* PURPOSE: Header file for ISAPI TPCC.DLL, defines structures and functions
used in the isapi tpcc.dll.
*
* Copyright Microsoft inc.1996, All Rights Reserved
*
* Author: PhilipDu, from tpcc.h by DamienL
* DamienL@ Microsoft.com
* philipdu@ Microsoft.com
*/
#ifndef _INC_TRANS
        #define _INC_TRANS
        #ifdef USE_ODBC
                #ifndef TIMESTAMP_STRUCT
                        #include <sqltypes.h>
                        #include <sql.h>
                        #include <sqlext.h>
                #endif
#else
        #ifndef _INC_SQLFRONT
                        #define DBNTWIN32
                        #include <sqlfront.h>
                        #include <sqldb.h>
                #endif
#endif
#ifndef DBINT
        typedef long DBINT;
#endif
#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];
		char dummy[600];
} STOCK_LEVEL_DATA;
#endif
```

tux.h

```
#ifndef TUX_H_INCLUDED
#define TUX_H_INCLUDED
#define SERVICE_CHARS 32
typedef union
{
		NEW_ORDER_DATA			NewOrderData;
		PAYMENT_DATA			PaymentData;
		ORDER_STATUS_DATA	OrderStatusData;
		DELIVERY_DATA			DeliveryData;
		STOCK_LEVEL_DATA	StockLevelData;
		char		ErrorMsg[ 4096]; // ack!!
} TRANS_DATA;
typedef struct
{
		int TermId;
		int SyncId;
		int bDeadlock;
		int bFailed;
		short DeadlockRetry;
		int Error;
		int Return;
// Note: Trans must be last
		TRANS_DATA Trans;
} TUX_DATA;

typedef struct
{
		char Service[ SERVICE_CHARS];
		// Note: Data must be last
		TUX_DATA Data;
} TUX_MSG;

// macros to compute the size of various bits of TUX_MSG. It is
// not enough to just add up the fields because of possible alignment
// issues
#define MSG_HEADER_SIZE( p)(( DWORD)((( char *)&( p) ->Data.Trans) - ((
char *)( p))))
#define NEW_ORDER_SIZE( p) (( MSG_HEADER_SIZE(( p)) + sizeof(
NEW_ORDER_DATA))
#define PAYMENT_SIZE( p) (( MSG_HEADER_SIZE(( p)) + sizeof(
PAYMENT_DATA))
#define ORDER_STATUS_SIZE( p) (( MSG_HEADER_SIZE(( p)) + sizeof(
ORDER_STATUS_DATA))
#define DELIVERY_SIZE( p) (( MSG_HEADER_SIZE(( p)) +  sizeof(
DELIVERY_DATA))
#define STOCK_LEVEL_SIZE( p) (( MSG_HEADER_SIZE(( p)) + sizeof(
STOCK_LEVEL_DATA))
#endif
```

Util.h

```
#ifndef TPCC_UTIL_H
		#define TPCC_UTIL_H
#endif
```

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
		Reqistry 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].blnUse )
					break;
			}
			if ( i < iQSlotts )
			{

		EnterCriticalSection(&DeliveryCriticalSection);
				pDeliveryCache[i].blnUse = 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].blnUse
= FALSE;
				i = iQSlotts;
				pDeliveryCache[i].blnUse = 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].blnUse =
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);

			DBSETLVERSION(login, DBVER60);		// due not
to convert numeric data type  to float values on server

			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\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);
				printf("deadlock_count %d\n");
			}
			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
```

error.c

```
#include <windows.h>
#include <string.h>
#include <stdio.h>
#include "trans.h"
#include "tpcc.h"
#include "util.h"
#include "error.h"
/* FUNCTION: void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int
iError, int iErrorType, char *szMsg)
*
* PURPOSE: This function displays an error message in the client browser.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intiErrorid of error message
* intiErrorTypeerror type, ERR_TYPE_SQL, ERR_TYPE_DBLIB, or
ERR_TYPE_WEBDLL
* intiTermIdterminal id from browser
* intiSyncidsync id from browser
* char *szMsgoptional error message string used with ERR_TYPE_SQL and
* ERR_TYPE_DBLIB
*
* RETURNS: None
*
* COMMENTS: If the error type is ERR_TYPE_WEBDLL the szmsg parameter
may be NULL because it
* is ignored.If the error type is ERR_TYPE_SQL or ERR_TYPE_DBLIB then the
szMsg
* parameter contains the text of the error message, so the szMsg parameter
cannot
* be NULL.
*
*/
void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType, char *szMsg, int iTermId, int iSyncId)
{
	int i;

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

		},
		{	ERR_COMMAND_UNDEFINED,
				"Command undefined."

				},
		{	ERR_NOT_IMPLEMENTED_YET,
				"Not Implemented Yet."

				},
		{	ERR_CANNOT_INIT_TERMINAL,
				"Cannot initialize client connection."

		},
		{	ERR_OUT_OF_MEMORY,
							"insufficient
memory."

		},
		{
		ERR_NEW_ORDER_NOT_PROCESSED,
		"Cannot process new Order form."

		},
		{	ERR_PAYMENT_NOT_PROCESSED,
				"Cannot process payment form."

				},
		{	ERR_NO_SERVER_SPECIFIED,
				"No Server name specified."

				},
		{
		ERR_ORDER_STATUS_NOT_PROCESSED,
		"Cannot process order status form."

		},
		{	ERR_W_ID_INVALID,
				"Invalid Warehouse ID."

					},
		{
		ERR_CAN_NOT_SET_MAX_CONNECTIONS,
		"Insufficient memory to allocate # connections."
				},
		{	ERR_NOSUCH_CUSTOMER,
				"No such customer."


		},
		{	ERR_D_ID_INVALID,
				"Invalid District ID Must be
1 to 10."
				},
		{	ERR_MAX_CONNECT_PARAM,
				"Max client connections
exceeded, run install to increase."		},
		{	ERR_INVALID_SYNC_CONNECTION,
		"Invalid Terminal Sync ID."

		},
		{	ERR_INVALID_TERMID,
				"Invalid Terminal ID."

					},
		{
		ERR_PAYMENT_INVALID_CUSTOMER,
		"Payment Form, No such Customer."

		},
		{	ERR_SQL_OPEN_CONNECTION,
				"SQLOpenConnection API Failed."

		},
		{
		ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,  "Stock Level
missing Threshold key \"TT*\"."
		},
		{
		ERR_STOCKLEVEL_THRESHOLD_INVALID,
		"Stock Level Threshold invalid data type range = 1 - 99."
		},
		{
		ERR_STOCKLEVEL_THRESHOLD_RANGE,
		"Stock Level Threshold out of range, range must be 1 - 99."
		},
		{
		ERR_STOCKLEVEL_NOT_PROCESSED,
		"Stock Level not processed."

		},
		{
		ERR_NEWORDER_FORM_MISSING_DID,
		"New Order missing District key \"DID*\"."
				},
		{
		ERR_NEWORDER_DISTRICT_INVALID,
		"New Order District ID Invalid range 1 - 10."
				},
		{
		ERR_NEWORDER_DISTRICT_RANGE,
		"New Order District ID out of Range. Range = 1 - 10."
		},
		{	ERR_NEWORDER_CUSTOMER_KEY,
				"New Order missing Customer key
\"CID*\"."
		},
		{
		ERR_NEWORDER_CUSTOMER_INVALID,
		"New Order customer id invalid data type, range = 1 to 3000."
		},
		{
		ERR_NEWORDER_CUSTOMER_RANGE,
		"New Order customer id out of range, range = 1 to 3000."
		},
		{
		ERR_NEWORDER_MISSING_IID_KEY,
		"New Order missing Item Id key \"IID*\"."
				},
		{
		ERR_NEWORDER_ITEM_BLANK_LINES,
		"New Order blank order lines all orders must be continuous."
		},
		{	ERR_NEWORDER_ITEMID_INVALID,
			"New Order Item Id is wrong data type, must be
numeric."		},
		{
		ERR_NEWORDER_MISSING_SUPPW_KEY,
		"New Order missing Supp_W key \"SP##*\"."
				},
		{	ERR_NEWORDER_SUPPW_INVALID,
			"New Order Supp_W invalid data type
must be numeric."		},
		{
		ERR_NEWORDER_MISSING_QTY_KEY,
		"New Order Missing Qty key \"Qty##*\"."
				},
		{	ERR_NEWORDER_QTY_INVALID,
			"New Order Qty invalid must be numeric
range 1 - 99."		},
		{	ERR_NEWORDER_SUPPW_RANGE,
```

```
                        "New Order Supp_W value out of range
range = 1 - Max Warehouses."            },
                        {               ERR_NEWORDER_ITEMID_RANGE,
                                        "New Order Item Id is out of range.
Range = 1 to 999999."                   },
                        {               ERR_NEWORDER_QTY_RANGE,
                                                "New Order Qty is out of
range. Range = 1 to 99."
                },
                        {               ERR_PAYMENT_DISTRICT_INVALID,
                        "Payment District ID is invalid must be 1 - 10."
                                        },
                        {
                ERR_NEWORDER_SUPPW_WITHOUT_ITEMID,
                "New Order Supp_W field entered without a corrisponding
Item_Id."       },
                        {
                ERR_NEWORDER_QTY_WITHOUT_ITEMID,
                "New Order Qty entered without a corrisponding Item_Id."
                        },
                        {
                ERR_NEWORDER_NOITEMS_ENTERED,
                "New Order Blank Items between items, items must be
continuous." },
                        {               ERR_PAYMENT_MISSING_DID_KEY,
                        "Payment missing District Key \"DID*\"."
                                                },
                        {               ERR_PAYMENT_DISTRICT_RANGE,
                                        "Payment District Out of range, range =
1 - 10."                                 },
                        {               ERR_PAYMENT_MISSING_CID_KEY,
                        "Payment missing Customer Key \"CID*\"."

                },
                        {
                ERR_PAYMENT_CUSTOMER_INVALID,
                "Payment Customer data type invalid, must be numeric."
                                        },
                        {               ERR_PAYMENT_MISSING_CLT,
                                        "Payment missing Customer Last
Name Key \"CLT*\"."
                },
                        {
                ERR_PAYMENT_LAST_NAME_TO_LONG,
                "Payment Customer last name longer than 16 characters."
                                        },
                        {
                ERR_PAYMENT_CUSTOMER_RANGE,
                        "Payment Customer ID out of range, must be 1 to
3000."                                  },
                        {               ERR_PAYMENT_CID_AND_CLT,
                                        "Payment Customer ID and Last Name
entered must be one or other."          },
                        {               ERR_PAYMENT_MISSING_CDI_KEY,
                        "Payment missing Customer district key \"CDI*\"."
                                                },
                        {               ERR_PAYMENT_CDI_INVALID,
                                        "Payment Customer district invalid must
be numeric."                            },
                        {               ERR_PAYMENT_CDI_RANGE,
                                                "Payment Customer
district out of range must be 1 - 10."  },
                        {               ERR_PAYMENT_MISSING_CWI_KEY,
                        "Payment missing Customer Warehouse key
\"CWI*\"."                                      },
                        {               ERR_PAYMENT_CWI_INVALID,
                                        "Payment Customer Warehouse invalid
must be numeric."                       },
                        {               ERR_PAYMENT_CWI_RANGE,
                                                "Payment Customer
Warehouse out of range, 1 to Max Warehouses."           },
                        {               ERR_PAYMENT_MISSING_HAM_KEY,
                        "Payment missing Amount key \"HAM*\"."

                },
                        {               ERR_PAYMENT_HAM_INVALID,
                                        "Payment Amount invalid data type
must be numeric."
                },
                        {               ERR_PAYMENT_HAM_RANGE,
                                                "Payment Amount out of
range, 0 - 9999.99."
                },
                        {
                ERR_ORDERSTATUS_MISSING_DID_KEY,
                "Order Status missing District key \"DID*\"."
                                                },
                        {               ERR_ORDERSTATUS_DID_INVALID,
                        "Order Status District invalid, value must be numeric
1 - 10."                },
                        {               ERR_ORDERSTATUS_DID_RANGE,
                                        "Order Status District out of range must
be 1 - 10."                             },
                        {
                ERR_ORDERSTATUS_MISSING_CID_KEY,
                "Order Status missing Customer key \"CID*\"."
                                                },
                        {
                ERR_ORDERSTATUS_MISSING_CLT_KEY,
                "Order Status missing Customer Last Name key \"CLT*\"."
                                        },
                        {               ERR_ORDERSTATUS_CLT_RANGE,
                                        "Order Status Customer last name
longer than 16 characters."             },
                        {               ERR_ORDERSTATUS_CID_INVALID,
                        "Order Status Customer ID invalid, range must be
numeric 1 - 3000."      },
                        {               ERR_ORDERSTATUS_CID_RANGE,
                                        "Order Status Customer ID out of range
must be 1 - 3000."                      },
                        {               ERR_ORDERSTATUS_CID_AND_CLT,
                                        "Order Status Customer ID and LastName entered
must be only one."      },
                        {               ERR_DELIVERY_MISSING_OCD_KEY,
                        "Delivery missing Carrier ID key \"OCD*\"."

                },
                        {               ERR_DELIVERY_CARRIER_INVALID,
                        "Delivery Carrier ID invalid must be numeric 1 - 10."
                                        },
                        {
                ERR_DELIVERY_CARRIER_ID_RANGE,
                "Delivery Carrier ID out of range must be 1 - 10."
                                        },
                        {               ERR_PAYMENT_MISSING_CLT_KEY,
                        "Payment missing Customer Last Name key
\"CLT*\"."                                      },
                        {               0,

                ""


                }
                };

                static char szNoMsg[] = "";
                char                    *szForm;

                if ( !szMsg  )
                        szMsg = szNoMsg;

                if ( iTermId > 0 && IsValidTermId(iTermId) )
                        szForm = Term.pClientData[iTermId].szBuffer; //if
termid valid use common terminal static buffer.
                else
                        szForm = Term.pClientData[0].szBuffer; //else term id
invalid so use common terminal static buffer.
                switch(iErrorType)
                {
                        case ERR_TYPE_WEBDLL:
                                for(i=0; errorMsgs[i].szMsg[0]; i++)
                                {
                                        if ( iError ==
errorMsgs[i].iError )
                                                break;
                                }
                                if ( !errorMsgs[i].szMsg[0] )
                                        i = 1;
                                strcpy(szForm,
                        "<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
                                wsprintf(szForm+strlen(szForm), "Error:
TPCCWEB(%d): %s", iError, errorMsgs[i].szMsg);
                                strcat(szForm,
"</FORM><BODY></HTML>");
                                WriteZString(pECB, szForm);
                                break;
                        case ERR_TYPE_SQL:
                                strcpy(szForm,
                        "<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
                                wsprintf(szForm+strlen(szForm), "Error:
SQLSVR(%d): %s", iError, szMsg);
                                strcat(szForm,
"</FORM><BODY></HTML>");
                                WriteZString(pECB, szForm);
                                break;
                        case ERR_TYPE_DBLIB:
                                strcpy(szForm,
                        "<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
                                wsprintf(szForm+strlen(szForm), "Error:
DBLIB(%d): %s", iError, szMsg);
                                strcat(szForm,
"</FORM><BODY></HTML>");
                                WriteZString(pECB, szForm);
                                break;
                        case ERR_TYPE_ODBC:
                                strcpy(szForm,
                        "<HTML><HEAD><TITLE>Welcome To TPC-
C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iErrorType);
                                wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
```

```
                                        wsprintf(szForm+strlen(szForm),
"<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
                                        wsprintf(szForm+strlen(szForm), "Error:
ODBC(%d): %s", iError, szMsg);
                                        strcat(szForm,
"</FORM><BODY></HTML>");
                                        WriteZString(pECB, szForm);
                                        break;
                }
                return;
}
```

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
```

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\\lnetinfo\\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;
}
```

pipe_routines.c

```
#include <windows.h>
#include <stdio.h>
#include "pipe_routines.h"
#include "trans.h"
#include "tpcc.h"
#include "tux.h"
const char *SERVER_PIPE_PATH = "\\\\.\\pipe\\tpcc_pipe.%d";
const char *CLIENT_PIPE_PATH = "\\\\.\\pipe\\tpcc_pipe.%d";

/*********************************************************
*
*		HANDLE OpenServerPipe(int PipeNumber, int TimeOut)
*
*********************************************************/
HANDLE 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.blnheritHandle= 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)
*
*********************************************************/
HANDLE OpenClientPipe(int ClientNumber)
{
		char		PipeName[_MAX_PATH];
		HANDLE		hPipe;
		DWORD		DesiredMode = PIPE_READMODE_MESSAGE;

		_snprintf(PipeName, sizeof(PipeName), CLIENT_PIPE_PATH,
ClientNumber);
		#ifdef _DEBUG
			fprintf(stderr, "OpenClientPipe begins for client %d\n",
ClientNumber);
		#endif
		while (1)
		{
			hPipe = CreateFile(PipeName, GENERIC_READ |
GENERIC_WRITE,
				FILE_SHARE_READ |
FILE_SHARE_WRITE,
				NULL,
				OPEN_EXISTING,
				FILE_ATTRIBUTE_NORMAL,
				0);
			if (hPipe != INVALID_HANDLE_VALUE)
				break;
			switch(GetLastError())
			{
				case ERROR_FILE_NOT_FOUND:
				// give the server a chance
					#ifdef _DEBUG
						fprintf(stderr,
"sleeping\n");
					#endif
					Sleep(20);
					break;
				default:
					fprintf(stderr,
"OpenClientPipe(%d): error in create of %s.Error = %d\n", ClientNumber,
PipeName, GetLastError());
					return
INVALID_HANDLE_VALUE;
					break;
			}
		}
		if (! SetNamedPipeHandleState(hPipe, &DesiredMode, NULL,
NULL))
		{
			fprintf(stderr, "OpenClientPipe(%d),
SetNamedPipeHandleStated faield in OpenclientPipe, error=%d\n",
ClientNumber, GetLastError());
			CloseHandle(hPipe);
			return INVALID_HANDLE_VALUE;
		}
		return hPipe;
}

/*********************************************************
*
*		BOOL ReadPipe
*
*********************************************************/
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
*
*****************************************************/
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;
}
```

sql_routines.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);
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* dbprocDBPROCESS id pointer
* intseverityseverity of error
* intdberrerror id
* intoserroperating system specific error code
* char* dberrstrprintable error description of dberr
* char* oserrstrprintable error description of oserr
*
* RETURNS: intINT_CONTINUEcontinue if error is SQLETIME else
INT_CANCEL action
*
* COMMENTS: None
*
*/
#ifndef USE_ODBC
int err_handler(DBPROCESS *dbproc, int severity, int
dberr, int oserr, char *dberrstr, char *oserrstr)
{
```

```
        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* dbprocDBPROCESS id pointer
* DBINTmsgnomessage number
* intmsgstatemessage state
* intseveritymessage severity
* char* msgtextprintable message description
*
* RETURNS: intINT_CONTINUEcontinue if error is SQLETIME else
INT_CANCEL action
* INT_CANCELcancel operation
*
* COMMENTS: This function also sets the dead lock dbproc variable if
necessary.
*
*/
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext)
{
        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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdterminal id of browser
* intiSyncIdsync id of browser
* DBPROCESS** dbprocpointer to returned DBPROCESS
* char* serverSQL server name
* char* databaseSQL server database
* char* useruser name
* char* passworduser password
* char* apppointer to returned application array
* int* spidpointer to returned spid
* long* pack_sizepointer to returned default pack size
*
* RETURNS: BOOLFALSEif successfull
* TRUEif an error occurs
*
* COMMENTS: None
*
*/
#ifdef USE_ODBC
        BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, int iSyncId, DBPROCESS ** dbproc, char *server, char
*database, char *user, char *password, char *app, int *spid)
        {
        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;
        PECBINFO   pEcbInfo;
        // 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);
        DBSETLVERSION(login, DBVER60);
        if ((*dbproc = dbopen(login, server )) == NULL)
                return TRUE;
        // set pECB data into dbproc
        pEcbInfo = (PECBINFO) malloc(sizeof(ECBINFO));
        pEcbInfo->bDeadlock = FALSE;
        pEcbInfo->pECB = pECB;
        pEcbInfo->iTermId = iTermId;
        pEcbInfo->iSyncId = iSyncId;
        dbsetuserdata(*dbproc, pEcbInfo);
        // Use the the right database
        dbuse(*dbproc, database);
        dbcmd(*dbproc, "select @@spid");
        dbsqlexec(*dbproc);
        while (dbresults(*dbproc) != NO_MORE_RESULTS)
        {
                dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *)
spid);
                while (dbnextrow(*dbproc) != NO_MORE_ROWS)
                ;
        }
        dbcmd(*dbproc, "set nocount on");
        dbsqlexec(*dbproc);
        while (dbresults(*dbproc) != NO_MORE_RESULTS)
        {
                while (dbnextrow(*dbproc) != NO_MORE_ROWS)
                ;
        }
        // rollback transaction on abort
        dbcmd(*dbproc, "set XACT_ABORT ON");
        dbsqlexec(*dbproc);
        while (dbresults(*dbproc) != NO_MORE_RESULTS)
        {
                while (dbnextrow(*dbproc) != NO_MORE_ROWS)
                ;
        }
        return FALSE;
}
#endif

/* FUNCTION: BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK
*pECB, DBPROCESS *dbproc)
*
* PURPOSE: This function closes the sql connection.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* DBPROCESS* dbprocpointer to DBPROCESS
*
* RETURNS: BOOLFALSEif successfull
* TRUEif an error occurs
*
* COMMENTS: None
*
*/
#ifdef USE_ODBC
BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB,
DBPROCESS *dbproc)
{
        if (dbproc )
        {
                SQLFreeStmt(dbproc->hstmt, SQL_DROP);
                SQLDisconnect(dbproc->hdbc);
                SQLFreeConnect(dbproc->hdbc);
                free(dbproc);
                dbproc = NULL;
        }
        return FALSE;
}
#else
BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB,
DBPROCESS *dbproc)
{
        if (dbclose(dbproc) == FAIL)
                return TRUE;
        return FALSE;
}
#endif

/* FUNCTION: SQLStockLevel(EXTENSION_CONTROL_BLOCK* pECB, int
iTermId, int iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA
*pStockLevel, short deadlock_retry)
*
* PURPOSE: This function handles the stock level transaction.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intiTermIdterminal id of browser
* intiSyncIdsync id of browser
* DBPROCESS* dbprocconnection db process id
* STOCK_LEVEL_DATA* pStockLevelstock level input / output data structure
* shortdeadlock_retryretry count if deadlocked
*
* RETURNS: BOOLFALSEif successfull
* TRUEif deadlocked
*
* COMMENTS: None
*
*/
#ifdef USE_ODBC
int SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
{
        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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdterminal id of browser
* intiSyncIdsync id of browser
* DBPROCESS* dbprocconnection db process id
* NEW_ORDER_DATA* pNewOrderpointer to new order structure for input/
output data
* shortdeadlock_retryretry count if deadlocked
*
* RETURNS: intTRUEtransaction committed
* FALSEitem number not valid
* -1deadlock max retry reached
*
*
* COMMENTS: None
*
*/
#ifdef USE_ODBC
int SQLNewOrder(EXTENSION_CONTROL_BLOCK* pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA *pNewOrder, short
deadlock_retry)
{
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;
if ((pEcbInfo = (PECBINFO) dbgetuserdata(dbproc)) )
{
pEcbInfo->pECB = pECB;
pEcbInfo->bFailed = FALSE;
pEcbInfo->iTermId = iTermId;
pEcbInfo->iSyncId = iSyncId;
}
pNewOrder->num_deadlocks = 0;
strcpy(tmpbuf, "tpcc_neworder");
for (tryit= 0; tryit < deadlock_retry; tryit++)
{
if (dbrpcinit(dbproc, tmpbuf, 0) == SUCCEED)
{
dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder->w_id);
dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder->d_id);
dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pNewOrder->c_id);
dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder-
>o_ol_cnt);
pNewOrder->o_all_local = 1;
for (i = 0; i < pNewOrder->o_ol_cnt; i++)
{
if (pNewOrder->o_all_local && pNewOrder->Ol[ i].ol_supply_w_id !=
pNewOrder->w_id )
pNewOrder->o_all_local = 0;
}
dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *)
&pNewOrder->o_all_local);
for (i = 0; i < pNewOrder->o_ol_cnt; i++)
{
dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *)
&pNewOrder->Ol[ i].ol_i_id);
dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *)
&pNewOrder->Ol[ i].ol_supply_w_id);
dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *)
&pNewOrder->Ol[ i].ol_quantity);
}
if (dbrpcexec(dbproc) == SUCCEED)
{
pNewOrder->total_amount= 0;
// Get resutls from order line
for (i = 0; i< pNewOrder->o_ol_cnt; i++)
{
if (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) &&
(rc != FAIL))
{
if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
{
while (dbnextrow(dbproc) != NO_MORE_ROWS)
{
if(pData= dbdata(dbproc, 1))
UtilStrCpy(pNewOrder->Ol[ i].ol_i_name, pData,
dbdatlen(dbproc, 1));
if(pData= dbdata(dbproc, 2))
pNewOrder->Ol[ i].ol_stock = (*(DBSMALLINT *) pData);
if(pData= dbdata(dbproc, 3))
UtilStrCpy(pNewOrder->Ol[ i].ol_brand_generic, pData,
dbdatlen(dbproc, 3));
if(pData= dbdata(dbproc, 4))
//pNewOrder->Ol[ i].ol_i_price = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pNewOrder->Ol[ i].ol_i_price, 8);
if(pData= dbdata(dbproc, 5))
//pNewOrder->Ol[ i].ol_amount = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pNewOrder->Ol[ i].ol_amount, 8);
pNewOrder->total_amount = pNewOrder->total_amount + pNewOrder->Ol[
i].ol_amount;
}
}
}
}
while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
{
if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
{
while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) &&
(rc != FAIL))
{
if(pData= dbdata(dbproc, 1))
//pNewOrder->w_tax = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pNewOrder->w_tax, 8);
if(pData= dbdata(dbproc, 2))
//pNewOrder->d_tax = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pNewOrder->d_tax, 8);
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);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pNewOrder->c_discount, 8);
if(pData= dbdata(dbproc, 6))
UtilStrCpy(pNewOrder->c_credit, pData, dbdatlen(dbproc, 6));
if(pData= dbdata(dbproc, 7))
{
datetime = *((DBDATETIME *) pData);
dbdatecrack(dbproc, &pNewOrder->o_entry_d, &datetime);
}
if(pData= dbdata(dbproc, 8)) commit_flag = (*(DBTINYINT*) pData);
}
}
}
}
}
if (SQLDetectDeadlock(dbproc))
{
pNewOrder->num_deadlocks++;
sprintf(printbuf," deadlock: retry: %d", pNewOrder->num_deadlocks);
Sleep(DEADLOCKWAIT* tryit);
}
else
{
if (commit_flag == 1)
{
pNewOrder->total_amount = pNewOrder->total_amount * ((1 + pNewOrder-
>w_tax + pNewOrder->d_tax) * (1 - pNewOrder->c_discount));
strcpy(pNewOrder->execution_status," Transaction commited.");
return TRUE;
}
else
{
strcpy(pNewOrder->execution_status," Item number is not valid.");
return FALSE;
}
}
}
// If we reached here, it means we quit after MAX_RETRY deadlocks
strcpy(pNewOrder->execution_status," Hit deadlock max. ");
return -1;//" deadlock max retry reached!"
}
#endif
/* FUNCTION: int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
deadlock_retry)
*
* PURPOSE: This function handles the payment transaction.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intiTermIdterminal id of browser
* intiSyncIdsync id of browser
* DBPROCESS* dbprocconnection db process id
* PAYMENT_DATA* pPaymentpointer to payment input/ output data structure
* shortdeadlock_retrydeadlock retry count
*
* RETURNS: intTRUEsuccess
* -1max deadlocked reached
*
* COMMENTS: None
*
*/
#ifdef USE_ODBC
int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
deadlock_retry)
{
int tryit;
char printbuf[ 25];
char buffer[ 255];
BOOL deadlock_detected;
PECBINFOp EcbInfo;
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;
if ((pEcbInfo = (PECBINFO) dbgetuserdata(dbproc)) )
{
pEcbInfo->pECB = pECB;
pEcbInfo->bFailed = FALSE;
pEcbInfo->iTermId = iTermId;
pEcbInfo->iSyncId = iSyncId;
}
pPayment->num_deadlocks = 0;
for (tryit= 0; tryit < deadlock_retry; tryit++)
{
if (dbrpcinit(dbproc, "tpcc_payment", 0) == SUCCEED)
{
dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pPayment->w_id);
dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pPayment->c_w_id);
dbrpcparam(dbproc, NULL, 0, SQLFLT8, -1, -1, (BYTE *) &pPayment-
>h_amount);
dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pPayment->d_id);
dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pPayment->c_d_id);
dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pPayment->c_id);
if (pPayment->c_id == 0)
{
dbrpcparam(dbproc, NULL, 0, SQLCHAR, -1, strlen(pPayment->c_last),
pPayment->c_last);
}
}
if (dbrpcexec(dbproc) == SUCCEED)
{
while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
{
if (DBROWS(dbproc) && (dbnumcols(dbproc) == 27))
{
while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
{
if(pData= dbdata(dbproc, 1))
pPayment->c_id = *((DBINT *) pData);
if(pData= dbdata(dbproc, 2))
UtilStrCpy(pPayment->c_last, pData, dbdatlen(dbproc, 2));
if(pData= dbdata(dbproc, 3))
{
datetime = *((DBDATETIME *) pData);
dbdatecrack(dbproc, &pPayment->h_date, &datetime);
}
if(pData= dbdata(dbproc, 4))
UtilStrCpy(pPayment->w_street_1, pData,
dbdatlen(dbproc, 4));
if(pData= dbdata(dbproc, 5))
UtilStrCpy(pPayment->w_street_2, pData,
dbdatlen(dbproc, 5));
if(pData= dbdata(dbproc, 6))
UtilStrCpy(pPayment->w_city, pData, dbdatlen(dbproc,
6));
if(pData= dbdata(dbproc, 7))
UtilStrCpy(pPayment->w_state, pData, dbdatlen(dbproc,
7));
if(pData= dbdata(dbproc, 8))
UtilStrCpy(pPayment->w_zip, pData, dbdatlen(dbproc, 8));
if(pData= dbdata(dbproc, 9))
UtilStrCpy(pPayment->d_street_1, pData, dbdatlen(dbproc, 9));
if(pData= dbdata(dbproc, 10))
UtilStrCpy(pPayment->d_street_2, pData, dbdatlen(dbproc, 10));
if(pData= dbdata(dbproc, 11))
UtilStrCpy(pPayment->d_city, pData, dbdatlen(dbproc, 11));
if(pData= dbdata(dbproc, 12))
UtilStrCpy(pPayment->d_state, pData, dbdatlen(dbproc, 12));
if(pData= dbdata(dbproc, 13))
UtilStrCpy(pPayment->d_zip, pData, dbdatlen(dbproc, 13));
if(pData= dbdata(dbproc, 14))
UtilStrCpy(pPayment->c_first, pData, dbdatlen(dbproc, 14));
if(pData= dbdata(dbproc, 15))
UtilStrCpy(pPayment->c_middle, pData, dbdatlen(dbproc, 15));
if(pData= dbdata(dbproc, 16))
UtilStrCpy(pPayment->c_street_1, pData, dbdatlen(dbproc, 16));
if(pData= dbdata(dbproc, 17))
UtilStrCpy(pPayment->c_street_2, pData, dbdatlen(dbproc, 17));
if(pData= dbdata(dbproc, 18))
UtilStrCpy(pPayment->c_city, pData, dbdatlen(dbproc, 18));
if(pData= dbdata(dbproc, 19))
UtilStrCpy(pPayment->c_state, pData, dbdatlen(dbproc, 19));
if(pData= dbdata(dbproc, 20))
UtilStrCpy(pPayment->c_zip, pData, dbdatlen(dbproc, 20));
if(pData= dbdata(dbproc, 21))
UtilStrCpy(pPayment->c_phone, pData, dbdatlen(dbproc, 21));
if(pData= dbdata(dbproc, 22))
{
datetime = *((DBDATETIME *) pData);
dbdatecrack(dbproc, &pPayment->c_since, &datetime);
}
if(pData= dbdata(dbproc, 23))
UtilStrCpy(pPayment->c_credit, pData, dbdatlen(dbproc, 23));
if(pData= dbdata(dbproc, 24))
//pPayment->c_credit_lim = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pPayment->c_credit_lim, 8);
if(pData= dbdata(dbproc, 25))
//pPayment->c_discount = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pPayment->c_discount, 8);
if(pData= dbdata(dbproc, 26))
//pPayment->c_balance = (*(DBFLT8 *) pData);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pPayment->c_balance, 8);
if(pData= dbdata(dbproc, 27))
```

```
UtilStrCpy(pPayment->c_data, pData, dbdatlen(dbproc, 27));
}
}
}
}
if (SQLDetectDeadlock(dbproc))
{
pPayment->num_deadlocks++;
sprintf(printbuf," deadlock: retry: %d", pPayment->num_deadlocks);
Sleep(DEADLOCKWAIT* tryit);
}
else
{
if (pPayment->c_id == 0 )
{
strcpy(pPayment->execution_status," Invalid Customer id, name.");
return 0;
}
else
strcpy(pPayment->execution_status," Transaction commited.");
return TRUE;
}
}
// If we reached here, it means we quit after MAX_RETRY deadlocks
strcpy(pPayment->execution_status," Hit deadlock max. ");
return -1; //" deadlock max retry reached!"
}
#endif
/* FUNCTION: int (EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
*
* PURPOSE: This function processes the Order Status transaction.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intiTermIdterminal id of browser
* intiSyncIdsync id of browser
* DBPROCESS* dbprocconnection db process id
* ORDER_STATUS_DATA* pOrderStatuspointer to Order Status data input/
output structure
* shortdeadlock_retrydeadlock retry count
*
* RETURNS: int- 1max deadlock reached
* 0No orders found for customer
* 1Transaction successfull
*
* COMMENTS: None
*
*/
#ifdef USE_ODBC
int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
{
int tryit;
int i;
BOOL not_done;
char buffer[ 255];
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);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pOrderStatus->OlOrderStatusData[ i].ol_amount, 8);
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);
dbconvert(dbproc, SQLNUMERIC, pData, sizeof(DBNUMERIC), SQLFLTN,
(CHAR *)&pOrderStatus->c_balance, 8);
if(pData= dbdata(dbproc, 8))
pOrderStatus->o_id = (*(DBINT *) pData);
}
}
if (i== 0)
return 0; //" No orders found for customer"
}
}
if (SQLDetectDeadlock(dbproc))
{
pOrderStatus->num_deadlocks++;
sprintf(printbuf," deadlock: retry: %d", pOrderStatus->num_deadlocks);
Sleep(DEADLOCKWAIT* tryit);
}
else
{
if (pOrderStatus->c_id == 0 && pOrderStatus->c_last[ 0]
== 0)
strcpy(pOrderStatus->execution_status," Invalid Customer id, name.");
else
strcpy(pOrderStatus->execution_status," Transaction commited.");
return 1;
}
}
// If we reached here, it means we quit after MAX_RETRY deadlocks
strcpy(pOrderStatus->execution_status," Hit deadlock max.");
return -1; //" deadlock max retry reached!"
}
#endif
/* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
*
* PURPOSE: This function checks to see if a sql server deadlock condition
exists.
*
* ARGUMENTS: DBPROCESS* dbprocconnection db process id to check
*
* RETURNS: BOOLFALSEno deadlock detected
* TRUEdeadlock condition exists
*
* COMMENTS: None
*
*/
BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
{
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: DBRPOCESSdbprocODBC dbprocess structure
*void *uPtrreturned data user pointer
*
* RETURNS: none
*
* COMMENTS: The caller is responsible for the contents of the uPtr.
*
*/
void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
{
dbproc->uPtr = uPtr;
}
/* FUNCTION: void dbsetuserdata(PDBPROCESS dbproc, void *uPtr)
*
* PURPOSE: This function returns the user pointer stored in a dbproc structure
*This functionality is not provided in odbc so this function
*provides it.
*
* ARGUMENTS: DBRPOCESSdbprocODBC dbprocess structure
*
* RETURNS: none
*
* COMMENTS: The returned pointer is placed in the dbproc structure by the
dbsetuserdata() API.
*
*/
void *dbgetuserdata(PDBPROCESS dbproc)
{
return dbproc->uPtr;
}
/* FUNCTION: void BindParameter(PDBPROCESS dbproc, UWORD ipar,
SWORD fCType, SWORD fSqlType, UDWORD cbColDef, SWORD ibScale,
PTR rgbValue, SDWORD cbValueMax)
*
* PURPOSE: This function wraps the functionality provided by the
SQLBindParameter
*allowing error process so that each bind call does not need to provide
*error and message checking.
*
* ARGUMENTS: PDBPROCESSdbprocpointer to odbc dbprocess structure
*UWORDiparParameter number, ordered sequentially left to right, starting at 1.
*SWORDfParamTypeThe type of the parameter.
*SWORDfCTypeThe C data type of the parameter.
*SWORDfSqlTypeThe SQL data type of the parameter.
*UDWORDcbColDefThe precision of the column or expression
*of the corresponding parameter marker.
*SWORDibScaleThe scale of the column or expression of the corresponding
*parameter marker.
*PTRrgbValueA pointer to a buffer for the parameter's data.
*SDWORDcbValueMaxMaximum length of the rgbValue buffer.
*void *uPtreturned data user pointer
*
* RETURNS: none
*
* COMMENTS: The returned pointer is placed in the dbproc structure by the
dbset
*
*/
void BindParameter(PDBPROCESS dbproc, UWORD ipar, SWORD fCType,
SWORD fSqlType, UDWORD cbColDef, SWORD ibScale, PTR rgbValue,
SDWORD cbValueMax)
{
RETCODE                 rc;
if (((PECBINFO) dbgetuserdata(dbproc))->bFailed )
return;
rc = SQLBindParameter(dbproc->hstmt, ipar, SQL_PARAM_INPUT, fCType,
fSqlType, cbColDef, ibScale, rgbValue, cbValueMax, NULL);
if (rc == SQL_ERROR)
ODBCError(dbproc);
return;
}
/* FUNCTION: void ODBCError(PDBPROCESS dbproc)
*
* PURPOSE: This function wraps the odbc error call so that the dblib
msg_handler is called.
*This allows the deadlock flag in the dbproc user data structure pEcbInfo in
*dbproc to be set if necessary.
*
* ARGUMENTS: DBRPOCESSdbprocODBC dbprocess structure
*
* RETURNS: none
*
* COMMENTS: none
*
*/
void ODBCError(PDBPROCESS dbproc)
{
```

```
SDWORD     lNativeError;
charszState[ 6];
charszMsg[ SQL_MAX_MESSAGE_LENGTH];
charszMsgText[ 256];
PECBINFOpEcbInfo;
charszTmp[ 256];
FILE* fp;
SYSTEMTIMEsystemTime;
pEcbInfo = (PECBINFO) dbgetuserdata(dbproc);
while(SQLError(henv, dbproc->hdbc, dbproc->hstmt,
szState, &lNativeError, szMsg, sizeof(szMsg), NULL) ==
SQL_SUCCESS )
{
msg_handler(dbproc, lNativeError, 0, 0, szMsg);
if (!lNativeError )
{
sprintf(szMsgText, "State = %s, %s", szState, szMsg);
ErrorMessage(pEcbInfo->pECB, -1, ERR_TYPE_ODBC,
szMsgText, pEcbInfo->iTermId, pEcbInfo->iSyncId);
pEcbInfo->bFailed = TRUE;
GetLocalTime(&systemTime);
fp = fopen(szErrorLogPath, "ab");
EnterCriticalSection(&ErrorLogCriticalSection);
sprintf(szTmp, "Error: SQLSVR(): %s", szMsg);
fprintf(fp, "%2.2d/%2.2d/%2.2d
%2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
systemTime.wYear, systemTime.wMonth, systemTime.wDay,
systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
szTmp);
LeaveCriticalSection(&ErrorLogCriticalSection);
fclose(fp);
}
}
return;
}
/* FUNCTION: BOOL ExecuteStatement(PDBPROCESS dbproc, szStatement)
*
* PURPOSE: This function wraps the odbc SQLExecDirect API so that error
handling and
*and deadlock are taken care of in a common location.
*
* ARGUMENTS: DBRPOCESSdbprocODBC dbprocess structure
*char* szStatementsql stored procedure statement to be executed.
*
* RETURNS: none
*
* COMMENTS: none
*
*/
BOOL ExecuteStatement(PDBPROCESS dbproc, char *szStatement)
{
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: DBRPOCESSdbprocODBC dbprocess structure
*UWORDicolColumn number of result data, ordered sequentially left to right,
starting at 1.
*SWORDfCTypeThe C data type of the result data. SQL_C_BINARY,
SQL_C_BIT, SQL_C_BOOKMARK,
*SQL_C_CHAR, SQL_C_DATE, SQL_C_DEFAULT, SQL_C_DOUBLE,
SQL_C_FLOAT, SQL_C_SLONG,
*SQL_C_SSHORT, SQL_C_STINYINT, SQL_C_TIME, SQL_C_TIMESTAMP,
SQL_C_ULONG,
*SQL_C_USHORT, SQL_C_UTINYINT, SQL_C_DEFAULT
*PTRrgbValuePointer to storage for the data.If rgbValue is a null pointer, the
*driver unbinds the column.
*SDWORDcbValueMaxMaximum length of the rgbValue buffer.For character
data, rgbValue
*must also include space for the null- termination byte.
* RETURNS: none
*
* COMMENTS: none
*
*/
BOOL BindColumn(PDBPROCESS dbproc, SQLUSMALLINT icol,
SQLSMALLINT fCType, SQLPOINTER rgbValue, SQLINTEGER cbValueMax)
{
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: DBRPOCESSdbprocODBC dbprocess structure
*
* RETURNS: none
*
* COMMENTS: none
*
*/
BOOL GetResults(PDBPROCESS dbproc)
{
PECBINFO pEcbInfo;
pEcbInfo = (PECBINFO) dbgetuserdata(dbproc);
if (pEcbInfo->bFailed )
return TRUE;
if (SQLFetch(dbproc->hstmt) == SQL_ERROR )
{
ODBCError(dbproc);
if (pEcbInfo->bDeadlock )
return FALSE;
return TRUE;
}
return FALSE;
}
/* FUNCTION: BOOL MoreResults(DBPROCESS dbproc)
*
* PURPOSE: This function wraps the odbc SQLMoreResults API so that error
handling and
*and deadlock are taken care of in a common location.
*
* ARGUMENTS: DBRPOCESSdbprocODBC dbprocess structure
*
* RETURNS: none
*
* COMMENTS: none
*
*/
BOOL MoreResults(PDBPROCESS dbproc)
{
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: DBRPOCESSdbprocODBC 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);
}
```

tpcc.c

```
/* FILE: TPCC.C
* Microsoft TPC- C Kit Ver.3.00.000
* Audited 08/ 23/ 96By Francois Raab
*
* Copyright Microsoft, 1996
*
* PURPOSE: Main module for TPCC.DLL which is an ISAPI service dll.
* Author: Philip Durr
* philipdu@ Microsoft.com
*/
#include <windows.h>
#include <process.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <io.h>
#include <fcntl.h>
#include "trans.h"// tpckit transaction header contains defination of structures
specific to TPC- C
#include "httpext.h"// ISAPI DLL information header
#include "tpcc.h"// this dlls specific structure, value e.t.header.
#include "sqlroutines.h"// the header files for the SQL routines (may be hiding
TUX)
#include "util.h"
#include "error.h"
#ifdef USE_ODBC
        HENVhenv;
#endif
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
        intbConnectionPooling = FALSE;
#endif
// allowable client command strings i.e.CMD= command
char    *szCmds[] =
{
        "..NewOrder..", "..Payment..", "..Delivery..", "..Order-Status..",
"..Stock-Level..", "..Exit..",
        "Submit", "Begin", "Process", "Menu", "Clear", "Users",
        ""
};

        // defined command string functions, called via CMD= command
http string from html client.
void (* DoCmd[])( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId) =
{
        NewOrderForm,
        PaymentForm,
        DeliveryForm,
        OrderStatusForm,
        StockLevelForm,
        Exitcmd,
        SubmitCmd,
        BeginCmd,
        ProcessCmd,
        MenuCmd,
        ClearCmd,
        NumberOfConnectionsCmd
};

// Terminal client id structure and interface defination
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: HANDLEhModulemodule handle
* DWORDul_reason_for_callreason for call
* LPVOIDlpReservedreserved for future use
*
* RETURNS: BOOLFALSEerrors occured in initialization
* TRUEDLL successfully initialized
*
* COMMENTS: None
*
*/
BOOL APIENTRY DllMain( HANDLE hModule, DWORD ul_reason_for_call,
LPVOID lpReserved)
{
        int                     i;
        static SECURITY_ATTRIBUTES sa;
        static PSECURITY_DESCRIPTOR pSD;
        switch( ul_reason_for_call )
        {
        case DLL_PROCESS_ATTACH:
                {
                        freopen("\\temp\\tpcc.log", "a", stderr);
                        setbuf( stderr, NULL);
                        fprintf( stderr, "logging started\n");
                }
                if ( ReadRegistrySettings() )
                {
                        MessageBox( NULL, "Cannot Find
TPCC Key in registry (run install.exe).", "Init", MB_OK | MB_ICONSTOP);
                        return FALSE;
                }
                InitializeCriticalSection(&CriticalSection);

                (*Term.Init)();

                if ( !(*Term.Allocate)() )
                {
                        MessageBox( NULL, "Error
Trm.Allocate().", "Init", MB_OK | MB_ICONSTOP);
                        return FALSE;
                }

                for( i= Term.iNext; i< Term.iAvailable; i++)
                        Term.pClientData[i].inUse = 0;
                Term.pClientData[0].inUse = 1;
                // create a security descriptor that allows anyone to
access the pipe...
                pSD = (PSECURITY_DESCRIPTOR)
malloc(SECURITY_DESCRIPTOR_MIN_LENGTH );
                if ( pSD == NULL )
                {
                        MessageBox( NULL, "Error malloc(
SECURITY_DESCRIPTOR_MIN_LENGTH)", "Init", MB_OK | MB_ICONSTOP);
                        return FALSE;
                }

                if ( !InitializeSecurityDescriptor( pSD,
SECURITY_DESCRIPTOR_REVISION) )
```

```
                {
                        MessageBox( NULL, "Error
InitializeSecurityDescriptor()", "Init", MB_OK | MB_ICONSTOP);
                        return FALSE;
                }
                // add a NULL disc.ACL to the security descriptor.
                if ( !SetSecurityDescriptorDacl( pSD, TRUE, (PACL)
NULL, FALSE) )
                {
                        MessageBox( NULL, "Error
SetSecurityDescriptorDacl().", "Init", MB_OK | MB_ICONSTOP);
                        return FALSE;
                }
                sa.nLength= sizeof( sa);
                sa.lpSecurityDescriptor= pSD;
                sa.bInheritHandle= TRUE;

                // open delivery named pipe...
                hPipe = CreateNamedPipe( lpszPipeName,
FILE_FLAG_OVERLAPPED | PIPE_ACCESS_DUPLEX,
                                        PIPE_TYPE_BYTE |
PIPE_READMODE_BYTE | PIPE_NOWAIT,
                                        1, 65535, 65535, 250,
&sa);

                if ( hPipe == INVALID_HANDLE_VALUE )
                {
                        MessageBox( NULL, "Error
CreateNamedPipe().", "Init", MB_OK | MB_ICONSTOP);
                        free( pSD);
                return FALSE;
                }

                bTpccExit = FALSE;
                if ( _beginthread( DeliveryDisconnect, 0, NULL ) == -1
)
                {
                        MessageBox( NULL, "Error
_beginthread()", "Init", MB_OK | MB_ICONSTOP);
                        return FALSE;
                }
                if (! SQLInit())
                        return FALSE;
                break;
        case DLL_THREAD_ATTACH:
                if (! SQLThreadAttach())
                        return FALSE;
                break;
        case DLL_THREAD_DETACH:
                if (! SQLThreadDetach())
                        return FALSE;
                break;
        case DLL_PROCESS_DETACH:
                if ( pSD )
                        free( pSD );
                bTpccExit = TRUE;
                if ( hPipe )
                        DisconnectNamedPipe( hPipe);
                if (hPipe != INVALID_HANDLE_VALUE )
                        CloseHandle( hPipe);
                (*Term.Restore)();
                SQLCleanup();
                DeleteCriticalSection(&CriticalSection);
                break;
        }
        return TRUE;
}

/* FUNCTION: void DeliveryDisconnect( void *ptr)
*
* PURPOSE: This function handles disconnecting the server side of the delivery
pipe when the
* delivery handler application shuts down.
*
* ARGUMENTS: void* ptrvoid pointer normally NULL passed from thread
handler.
*
* RETURNS: None
*
* COMMENTS: This function runs as thread which allows the client pipe to
disconnect by
* sending a byte back though the pipe to the server i.e.this DLL.
*/
static void DeliveryDisconnect( void *ptr)
{
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* pVerpassed in structure in which to
place expected version number.
*
* RETURNS: TRUEinet service expected return value.
*
* COMMENTS: None
*
*/
BOOL WINAPI GetExtensionVersion( HSE_VERSION_INFO *pVer)
{
        pVer->dwExtensionVersion = MAKELONG(
HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
        lstrcpyn( pVer->lpszExtensionDesc, "TPC-C Server.",
HSE_MAX_EXT_DLL_NAME_LEN);
        return TRUE;
}

/* FUNCTION: DWORD WINAPI HttpExtensionProc(
EXTENSION_CONTROL_BLOCK *pECB)
*
* PURPOSE: This function is the main entry point for the TPCC DLL.The internet
service
* calls this function passing in the http string.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBstructure pointer to
passed in internet
* service information.
*
* RETURNS: DWORDHSE_STATUS_SUCCESSconnection can be dropped if
error
* HSE_STATUS_SUCCESS_AND_KEEP_CONNkeep connect valid comment
sent
*
* COMMENTS: None
*
*/

DWORD WINAPI HttpExtensionProc( EXTENSION_CONTROL_BLOCK *pECB)
{
        int iCmd, FormId, TermId, iSyncId;
        FILE *fp;
        if ( iMaxConnections == -1 )
        {
                ErrorMessage( pECB,
ERR_CAN_NOT_SET_MAX_CONNECTIONS, ERR_TYPE_WEBDLL, NULL, -1,
-1);
                return HSE_STATUS_SUCCESS;
        }
  // if registry setting is for html logging then show http string passed in.


        if ( bLog )
        {
                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) )
                {
                        ErrorMessage( pECB,
ERR_INVALID_TERMID, ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
                        return
HSE_STATUS_SUCCESS_AND_KEEP_CONN;
                }
                // must have a valid syncid here since termid is valid
                if ( iSyncId < 1 || iSyncId !=
Term.pClientData[TermId].iSyncId )
                {
                        ErrorMessage( pECB,
```

```
ERR_INVALID_SYNC_CONNECTION, ERR_TYPE_WEBDLL, NULL, TermId,
iSyncId);
                                        return
HSE_STATUS_SUCCESS_AND_KEEP_CONN;
                                }
                }
                // set use time
                Term.pClientData[TermId].iTickCount = GetTickCount();
                // go execute http: command
                (*DoCmd[iCmd])( pECB, FormId, TermId, iSyncId);
                // finish up and keep connection
                return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}

/* FUNCTION: static BOOL IsValidTermId( int TermId)
*
* PURPOSE: This function checks to see of the passed in terminal id is valid.
*
* ARGUMENTS: intTermIdclient terminal id
*
* RETURNS: BOOLFALSETerminal ID Invalid
* TRUETerminal ID valid
*
* COMMENTS: None
*
*
*/
BOOL IsValidTermId(int TermId)
{
                return (BOOL) ( TermId > 0 && TermId <= Term.iAvailable &&
Term.pClientData[TermId].inUse );
}

/* FUNCTION: BOOL ProcessQueryString( EXTENSION_CONTROL_BLOCK
*pECB, int
*pCmd, int *pFormId, int *pTermId, int *pSyncId)
*
* PURPOSE: This function extracts the relevent information out of the http
command passed in from
* the browser.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBstructure pointer to
passed in internet
* service information.
* int* pCmdreturned command id
* int* pFormIdreturned active form client browser is on
* int* pTermIdreturned client terminal id
*
* RETURNS: BOOLFALSEsuccess
* TRUEcommand passed in is invalid
*
* COMMENTS: If this is the initial connection i.e.client is at welcome screen
then
* there will not be a terminal id or current form id if this is the case
* then the pTermid and pFormid return values are undefined.
*/
BOOL ProcessQueryString( EXTENSION_CONTROL_BLOCK *pECB, int
*pCmd, int *pFormId, int *pTermId, int *pSyncId)
{
                char *ptr;
                char szBuffer[25];
                char szTmp[25];
                char *dest = szBuffer;
                int i;
                if ( (ptr = strstr( pECB->lpszQueryString, "FORMID=")) )
                                *pFormId = *( ptr+ 7) &0x0F;
                if ( (ptr = strstr( pECB->lpszQueryString, "TERMID=")) )
                {
                                *pTermId = atoi(( ptr+ 7));
                                if ( *pTermId == 0 )// terminal id 0 used internally
                                                *pTermId = -1;
                                if ( *pTermId == -2 )// login screen
                                                *pTermId = 0;
                }
                else
                                *pTermId = 0;
                                if ( (ptr = strstr( pECB->lpszQueryString, "SYNCID="))
)
                                                *pSyncId = atoi(( ptr+ 7));
                                else
                                                *pSyncId = 0;
                                if ( !( ptr = strstr( pECB->lpszQueryString, "CMD=")) )
                                {
                                                ptr = szBuffer;
                                                if ( !stricmp( szBuffer, "Default") )
                                                                strcpy( szBuffer,
"CMD=Begin");
                                                switch( *pFormId )
                                                {
                                                                case WELCOME_FORM:
                                                                                strcpy(
szBuffer, "CMD=Submit");
                                                                                break;
                                                                case
MAIN_MENU_FORM:
                                                                                strcpy(
szBuffer, "CMD=NewOrder");
                                                                                break;
                                                                case
NEW_ORDER_FORM:
                                                                case PAYMENT_FORM:
                                                                case DELIVERY_FORM:
                                                                case
ORDER_STATUS_FORM:
                                                                case
STOCK_LEVEL_FORM:
                                                                                if (
!(*pTermId) )

                return FALSE;
                                                                                if (
GetKeyValue( pECB->lpszQueryString, "PI*", szTmp, sizeof( szTmp)) )

                strcpy( szBuffer, "CMD=Process");
                                                                                else
                                                                                {

                strcpy( szBuffer, "CMD=");

                strcat( szBuffer, szCmds[*pFormId - NEW_ORDER_FORM]);
                                                                                }
                                                                                break;
                                                                default:
                                                                                return
FALSE;
                                                }
                                }
                                ptr += 4;
                                while( *ptr && *ptr != '&' )
                                                *dest++ = *ptr++;
                                                *dest = 0;
                                                for( i= 0; szCmds[i][0]; i++)
                                                                {
if ( !strcmp( szCmds[i], szBuffer) )
{
*pCmd = i;
return TRUE;
}
}
return FALSE;
}

/* FUNCTION: void NewOrderForm( EXTENSION_CONTROL_BLOCK *pECB,
int iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function wraps the functionality needed for the TPC- C New
Order Form.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passe in internet
* service information.
*
* RETURNS: None
*
* COMMENTS: None
*
*/
void NewOrderForm( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
                WriteZString( pECB, MakeNewOrderForm( iTermId, iSyncId,
TRUE, FALSE));
                UNUSEDPARAM( iFormId);
                return;
}

/* FUNCTION: void PaymentForm( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function wraps the functionality needed for the TPC- C
Payment Form.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void PaymentForm( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
{
WriteZString( pECB, MakePaymentForm( iTermId, iSyncId,
TRUE) );
UNUSEDPARAM( iFormId);
}
/* FUNCTION: void DeliveryForm( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function wraps the functionality needed for the TPC- C
Delivery Form.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void DeliveryForm( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
{
WriteZString( pECB, MakeDeliveryForm( iTermId, iSyncId, TRUE, TRUE) );
UNUSEDPARAM( iFormId);
}
/* FUNCTION: void OrderStatusForm( EXTENSION_CONTROL_BLOCK
```

```
*pECB, int iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function wraps the functionality needed for the TPC- C Order
Status Form.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling borwser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void OrderStatusForm( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
WriteZString( pECB, MakeOrderStatusForm( iTermId, iSyncId, TRUE) );
UNUSEDPARAM( iFormId);
}
/* FUNCTION: void StockLevelForm( EXTENSION_CONTROL_BLOCK
*pECB, int iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function wraps the functionality needed for the TPC- C Stock
Level Form.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void StockLevelForm( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
WriteZString( pECB, MakeStockLevelForm( iTermId, iSyncId, TRUE) );
return;
}
/* FUNCTION: void Exitcmd( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function removes a terminal id from use, the allocated
structure however remains
* valid so the next request for a new client will not require a new memory
allocation.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void Exitcmd( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId,
int iSyncId)
{
(*Term.Delete)( pECB, iTermId);
WriteZString( pECB, MakeWelcomeForm() );
UNUSEDPARAM( iFormId);
UNUSEDPARAM( iSyncId);
return;
}
/* FUNCTION: void SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function allocated a new terminal id in the Term structure
array.
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: A terminal id can be allocated but still be invalid if the requested
warehouse number
* is outside the range specified in the registry.This then will force the client id
* to be invalid and an error message sent to the users browser.
*/
void SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
        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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: SQL server must be specified, however the user and password
parameters are optional.
* The complete command line is CMD= Begin&Server= server&User=
sa&Psw=&.The &are used
* to separate parameters which is internet browser standard.
*/
void BeginCmd( EXTENSION_CONTROL_BLOCK *pECB, int iFormId,int
iTermId, int iSyncId)
{
        LPSTR pQueryString;
        pQueryString = pECB->lpszQueryString;
        if ( !GetKeyValue( pQueryString, "Server", szServer, sizeof(
szServer)) )
        {
                ErrorMessage( pECB,
ERR_NO_SERVER_SPECIFIED, ERR_TYPE_WEBDLL, NULL, iTermId,
iSyncId);
                return;
        }
        if ( !GetKeyValue( pQueryString, "User", szUser, sizeof( szUser)) )
                strcpy( szUser, "sa");
        if ( !GetKeyValue( pQueryString, "Psw", szPassword, sizeof(
szPassword)) )
                strcpy( szPassword, "");
        if ( !GetKeyValue( pQueryString, "Db", szDatabase, sizeof(
szDatabase)) )
                strcpy( szDatabase, "tpcc");
        WriteZString( pECB, MakeWelcomeForm() );
        UNUSEDPARAM( iFormId);
        return;
}
/* FUNCTION: void ProcessCmd( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function process the passed in http command
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void ProcessCmd( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
        switch( iFormId )
        {
        case WELCOME_FORM:
                return;
        case MAIN_MENU_FORM:
                return;
        case NEW_ORDER_FORM:
                ProcessNewOrderForm( pECB, iTermId, iSyncId);
                return;
        case PAYMENT_FORM:
                ProcessPaymentForm( pECB, iTermId, iSyncId);
                return;
        case DELIVERY_FORM:
                ProcessDeliveryForm( pECB, iTermId, iSyncId);
                return;
        case ORDER_STATUS_FORM:
                ProcessOrderStatusForm( pECB, iTermId, iSyncId);
                return;
        case STOCK_LEVEL_FORM:
                ProcessStockLevelForm( pECB, iTermId, iSyncId);
                return;
        }
}

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

```
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: Use this function with caution, it may cause unpredictable
results
* if existing browsers attempt to use the web client with out
* beginning at the login screen for each client.
*/
void ClearCmd( EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
int i;
EnterCriticalSection(&CriticalSection);
for( i= 0; i< Term.iAvailable; i++)
{
if ( Term.pClientData[i].inUse )
(*Term.Delete)( pECB, i);
}
Term.iNext= 0;
Term.iAvailable= 0;
Term.iMasterSyncId= 1;
if ( Term.pClientData )
free( Term.pClientData);
Term.pClientData= NULL;
Term.bInit= FALSE;
(*Term.Init)();
if ( !(*Term.Allocate)() )
{
ErrorMessage( pECB, ERR_MAX_CONNECT_PARAM, ERR_TYPE_WEBDLL,
NULL, iTermId, iSyncId);
return;
}
for( i= Term.iNext; i< Term.iAvailable; i++)
Term.pClientData[i].inUse = 0;
Term.pClientData[0].inUse = 1;
LeaveCriticalSection(&CriticalSection);
WriteZString( pECB, MakeWelcomeForm() );
return;
}
/* FUNCTION: void MenuCmd( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function causes an exit to the main menu
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void MenuCmd( EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
{
WriteZString( pECB, MakeMainMenuForm( iTermId, iSyncId) );
return;
}
/* FUNCTION: void NumberOfConnectionsCmd(
EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function returns to the browser the total number of active
terminal ids
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*/
void NumberOfConnectionsCmd( EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
{
int i;
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* pECBpassed in structure
pointer from inetsrv.
* char* szStrstring to display in the client browser.
*
* RETURNS: None
*
* COMMENTS: This function assumes that the string to written to the client
browser has
* been formatted in an HTML manner.
*/
void WriteZString( EXTENSION_CONTROL_BLOCK *pECB, char *szStr)
{
        FILE* fp;
        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* pECBpassed in structure
pointer from inetsrv.
* char* formatprintf style format string
* ...other arguments as required by printf style format string.
*
* RETURNS: None
*
* COMMENTS: This function is mainly used for developmental support.
*/
static void h_printf( EXTENSION_CONTROL_BLOCK *pECB, char *format, ...)
{
char szBuff[512];
char szTmp[512];
va_list marker;
va_start( marker, format );
vsprintf( szTmp, format, marker);
va_end( marker );
wsprintf( szBuff, "<html>%s</html>", szTmp) + 1;
WriteZString( pECB, szBuff);
return;
}
/* FUNCTION: BOOL GetKeyValue( char *pQueryString, char
*pKey, char *pValue, int iMax)
*
* PURPOSE: This function parses a http formatted string for specific key values.
*
* ARGUMENTS: char* pQueryStringhttp string from client browser
* char* pKeykey value to look for
* char* pValuecharacter array into which to place key's value
* intiMaxmaximum length of key value array.
*
* RETURNS: BOOLFALSEkey value not found
* TRUEkey valud found
*
*
* COMMENTS: http keys are formatted either KEY= value&or KEY= value\
0.This DLL formats
* TPC- C input fields in such a manner that the keys can be extracted in the
* above manner.
*/
static BOOL GetKeyValue( char *pQueryString, char *pKey, char *pValue, int
iMax)
{
char *ptr;
if ( !( ptr= strstr( pQueryString, pKey)) )
return FALSE;
if ( !( ptr= strchr( ptr, '=')) )
return FALSE;
ptr++;
iMax--;
while( *ptr && *ptr != '&' && iMax)
{
*pValue++ = *ptr++;
iMax--;
}
*pValue = 0;
return TRUE;
}

/* FUNCTION: void TermInit( void)
*
* PURPOSE: This function initializes the client ternimal structure it is called
when the TPCC.DLL
* is first loaded by the inet service.
```

```
*
* ARGUMENTS: none
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void TermInit( void)
{
		if ( Term.bInit )
		return;
		Term.iNext= 0;
		Term.iMasterSyncId= 1;
		Term.iAvailable= 0;
		Term.pClientData= NULL;
		Term.bInit= TRUE;
		return;
}

/* FUNCTION: void TermRestore( void)
*
* PURPOSE: This function frees allocated resources associated with the
terminal structure.
*
* ARGUMENTS: none
*
* RETURNS: None
*
* COMMENTS: This function is called only with the inet service unloads the
TPCC.DLL
*
*/
static void TermRestore( void)
{
Term.iNext= 0;
Term.iAvailable= 0;
Term.iMasterSyncId= 0;
if ( Term.pClientData )
free( Term.pClientData);
Term.pClientData= NULL;
Term.bInit= FALSE;
return;
}
/* FUNCTION: int TermAllocate( void)
*
* PURPOSE: This function allocates more terminal array entries in the Term
structure.
*
* ARGUMENTS: None
*
* RETURNS: intTRUE or 1 if sucessfull
* intFALSE or 0 if terminal id cannot be allocated.
*
* COMMENTS: None
*
*/
static int TermAllocate( void)
{
		Term.iAvailable += 32;
		if ( !Term.pClientData )
			Term.pClientData = (PCLIENTDATA) malloc(
Term.iAvailable * sizeof( CLIENTDATA));
		else
			Term.pClientData =
				(PCLIENTDATA) realloc(
Term.pClientData, Term.iAvailable * sizeof( CLIENTDATA));
		return ( Term.pClientData ) ? 1 : 0;
}

/* FUNCTION: int TermAdd( EXTENSION_CONTROL_BLOCK *pECB,char
*pQueryString)
*
* PURPOSE: This function assigns a terminal id which is used to identify a
client browser.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* char* pQueryStringhttp query string passed to this DLL.
*
* RETURNS: 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, iTotalConnections,
iTickCount;

		EnterCriticalSection(&CriticalSection);
		for( i= 0, iTotalConnections = 0; i< Term.iAvailable; i++)
		{
			if ( Term.pClientData[i].inUse )
				iTotalConnections++;
		}
		if ( iTotalConnections >= iMaxConnections )
		{
			for( iCurrent = 1, i= 1, iTickCount = 0x7FFFFFFF; i<
iMaxConnections; i++)
			{
				if ( iTickCount >
Term.pClientData[i].iTickCount )
				{
					iTickCount =
Term.pClientData[i].iTickCount;
					iCurrent = i;
				}
			}
		}
		else
		{
			for( i= 0; i< Term.iAvailable; i++)
			{
				if ( !Term.pClientData[i].inUse )
				break;
			}
			iCurrent = i;
		}
		if ( i == Term.iAvailable )
		{
			Term.iNext = Term.iAvailable;
			if ( !(*Term.Allocate)() )
				goto TermAddErr1;
			for( i= Term.iNext; i< Term.iAvailable; i++)
				Term.pClientData[i].inUse = 0;
			iCurrent = Term.iNext;
		}
		Term.pClientData[iCurrent].inUse = 1;
		if ( !GetKeyValue( pQueryString, "w_id", szTmp, sizeof( szTmp)) )
			goto TermAddErr1;
		Term.pClientData[iCurrent].w_id = (short) atoi( szTmp);
		if ( !GetKeyValue( pQueryString, "d_id", szTmp, sizeof( szTmp)) )
			goto TermAddErr1;
		Term.pClientData[iCurrent].d_id = atoi( szTmp);
		Term.pClientData[iCurrent].iTickCount = GetTickCount();
		Term.pClientData[iCurrent].iSyncId = Term.iMasterSyncId++;
		if ( Init( pECB, iCurrent,Term.pClientData[iCurrent].iSyncId,
szServer, szUser, szPassword, szDatabase) )
		{
			(*Term.Delete)( pECB, iCurrent);
			goto TermAddErr1;
		}
		LeaveCriticalSection(&CriticalSection);
		return iCurrent;
		TermAddErr1:
		LeaveCriticalSection(&CriticalSection);
		return -1;// terminal unsuccessfully added
}

/* FUNCTION: void TermDelete( EXTENSION_CONTROL_BLOCK *pECB, int id)
*
* PURPOSE: This function makes a terminal entry in the Term array available for
reuse.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intidTerminal id of client exiting
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void TermDelete( EXTENSION_CONTROL_BLOCK *pECB, int id)
{
if ( id >= 0 && id < Term.iAvailable )
{
Close( pECB, id, -1);
Term.pClientData[id].inUse = 0;
}
return;
}

/* FUNCTION: BOOL Init( EXTENSION_CONTROL_BLOCK *pECB, int iTermId,
int iSyncId, char *szServer, char *szUser, char *szPassword, char *szDatabase)
*
* PURPOSE: This function initializes the sql connection for use.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intiTermIdid of browser client that this connection is for.
* intiSyncIdsync id for this client session
* char* szServersql server name
* char* szUseruser name
* char* szPassworduser password
* char* szDatabasedatabase to use
*
* RETURNS: BOOLFALSEif successfull
* TRUEif an error occurs and connection cannot be established.
*
* COMMENTS: None
*
*/
BOOL Init( EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
char *szServer, char *szUser, char *szPassword,char *szDatabase)
{
		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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdid of browser client that this connection is for.
* intiSyncIdsync id of client browser
*
* RETURNS: BOOLFALSEif successfull
* TRUEif an error occurs and connection cannot be terminated.
*
* COMMENTS: None
*
*/
static BOOL Close( EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId)
{
PECBlNFO   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* szDestDestination buffer where formatted string is to be
placed
* char* szPicpicture string which describes how character value is to be
* formatted.
* char* szSrccharacter string value.
*
* RETURNS: None
*
* COMMENTS: This functions is used to format TPC- C phone and zip value
strings.
*
*/
static void FormatString( char *szDest, char *szPic, char *szSrc)
{
while( *szPic )
{
if ( *szPic == 'X' )
{
if ( *szSrc )
*szDest++ = *szSrc++;
else
*szDest++ = ' ';
}
else
*szDest++ = *szPic;
szPic++;
}
*szDest = 0;
return;
}


/* FUNCTION: char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL blnput)
 *
 * PURPOSE:             This function constructs the Stock Level HTML page.
 *
 * ARGUMENTS:           int
                iTermId     client browser terminal id
 *                                                      int
                iSyncId     client browser sync id
 *                                                      BOOL
                blnput      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 blnput)
{
                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 ( blnput )
                                strcat(szForm, "<INPUT TYPE=\"hidden\"
NAME=\"PI*\" VALUE=\"\">");
                strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">");
                wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", STOCK_LEVEL_FORM);
                wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
                wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
                strcat(szForm, "<PRE>                           Stock-Level<BR>");
                wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d   District:
%2.2d<BR><BR>", Term.pClientData[iTermId].stockLevelData.w_id,
Term.pClientData[iTermId].stockLevelData.d_id);
                if ( blnput )
                {
                                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);
		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);
		}

		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=\"\">");

        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=\"\">");

	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].o
l_supply_w_id,

			Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].o
l_i_id,

			Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].o
l_quantity,

			Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].o
l_amount,

			Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].o
l_delivery_d.day,

			Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].o
l_delivery_d.month,

			Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].o
l_delivery_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)
 *
 * 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 *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput)
{
	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\">");
	wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">", DELIVERY_FORM);
	wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
	wsprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);

	strcat(szForm,		"<PRE>
Delivery<BR>" );

	wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d<BR><BR>",
Term.pClientData[iTermId].deliveryData.w_id);

	if ( bInput )
		strcat( szForm, "Carrier Number: <INPUT
NAME=\"OCD*\" SIZE=1><BR><BR>");
```

```
		else
		{
			wsprintf(szForm+strlen(szForm), "Carrier Number:
%2.2d<BR><BR>",

		Term.pClientData[iTermId].deliveryData.o_carrier_id);
		}
		if ( bInput )
		{
			strcat( szForm, "Execution Status:<BR></PRE>"

		"<HR><INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Process\">"

		"<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">" );
		}
		else
		{
			wsprintf(szForm+strlen(szForm), "Execution Status:
%25.25s<BR></PRE>",

		Term.pClientData[iTermId].deliveryData.execution_status);

			strcat(szForm,		"<HR><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">"

		"<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..Payment..\">"

		"<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"..Delivery..\">"

		"<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Order-
Status..\">"

		"<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Stock-
Level..\">"

		"<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..Exit..\">" );
		}

		strcat( szForm,		"</BODY></FORM></HTML>" );

		return szForm;
}


/* FUNCTION: void ProcessNewOrderForm( EXTENSION_CONTROL_BLOCK*
pECB, int iTermId, int iSyncId)
*
* PURPOSE: This function gets and validates the input data from the new order
form
* filling in the required input variables.it then calls the SQLNewOrder
* transaction, constructs the output form and writes it back to client
* browser.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBpassed in structure
pointer from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncId client browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessNewOrderForm( EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
		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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncId client browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessPaymentForm( EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
		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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncId client browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessOrderStatusForm( EXTENSION_CONTROL_BLOCK *pECB,
int iTermId, int iSyncId)
{
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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncIdclinet browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessDeliveryForm( EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
        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* pECBpassed in structure
pointer from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessStockLevelForm( EXTENSION_CONTROL_BLOCK *pECB,
int iTermId, int iSyncId)
{
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 );

//			pPaymentData->c_id = 10; //-------- add oba ---------//
			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;
//
                        pOrderStatusData->c_id = 10;

            }
            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;
            iDelayMs                = 100;
            iDeadlockRetry          = (short)3;
            strcpy(szTpccLogPath, "tpcclog.");


            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, "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;

            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 );
}

#ifdef USE_ODBC
                /* FUNCTION: static int ODBCError(DBPROCESS *dbproc)
                 *
                 * PURPOSE:             This function Handles the processing of
errors from ODBC APIs
                 *
                 * ARGUMENTS:           PDBPROCESS
                 *
                 * RETURNS:                             BOOL        FALSE
                if string is not all numeric
                 *
                TRUE        if string contains only numeric characters i.e. '0' - '9'
                 *
                 * COMMENTS:            None
                 *
                 */

                static int ODBCError(DBPROCESS *dbproc)
                {
                        RETCODE     rc;
                        SDWORD      lNativeError;
                        BOOL                    bError;
                        char        szState[6];
                        char        szMsg[SQL_MAX_MESSAGE_LENGTH];
                        char                    timebuf[128];
                        char                    datebuf[128];

                        pdbproc->deadlock_detected = TRUE;
                        bError = FALSE;

                        while( SQLError(dbproc->henv,
                                dbproc->hdbc,
                                dbproc->hstmt,
                                szState,
                                &lNativeError,
                                szMsg,
                                sizeof(szMsg),
                                NULL) != SQL_NO_DATA_FOUND )
                                {
                                        if (lNativeError == 1205)
                                                dbproc-
>deadlock_detected = TRUE;
                                        else
                                        {
                                                _strtime(timebuf);
                                                _strdate(datebuf);

                                                h_printf(dbproc->pECB,
"%s %s : ODBC Error:  State=%s, Error=%ld, %s\n", datebuf, timebuf, szState,
lNativeError, szMsg);
                                                bError = TRUE;
                                        }
                                }
                                if ( bError )
                                        return -1;
                                return dbproc->deadlock_detected;
                }
#endif

/* 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;
}
```

tux_client.c

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <direct.h>
#include "atmi.h" /* TUXEDO Header File */
#ifdef USE_ODBC
                #include <sqltypes.h>
                #include <sql.h>
                #include <sqlext.h>
                HENV henv;
#else
                #define         DBNTWIN32
                #include <sqlfront.h>
                #include <sqldb.h>
#endif
#include "trans.h"
#include "tpcc.h"
#include "pipe_routines.h"
#include "tux.h"
```

```
#define  SERVICE_BUF_SIZE              16
typedef char *EXTENSION_CONROL_BLOCK;
const int TIMEOUT = 1000* 30; // timeout in milliseconds
const int ARGSIZE = 1024;
const char *LOG_PATH="c:\\temp\\tux_logs\\";
const char *LOG_NAME="client_%d.txt";
// Global variables set as parameters
int                         InitialCreate  = 0;
int                         ClientNumber                = 0;
char        *TuxBuffer;

//------@add -------//
char        *TuxBuffer_no;
char        *TuxBuffer_pay;
char        *TuxBuffer_os;
char        *TuxBuffer_stock;

/*********************
*
*           BOOL TuxInit()
*
*********************/
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();
                        }
/*
                        TuxBuffer_no = (char *) tpalloc("CARRAY", NULL,
sizeof(TUX_MSG));
                        if (TuxBuffer_no != NULL)
                                    bReturn = TRUE;
                        else
                        {
                                    fprintf(stderr, "tpalloc of buffer failed\n");
                                    tpterm();
                        }
                        TuxBuffer_pay = (char *) tpalloc("CARRAY", NULL,
sizeof(TUX_MSG));
                        if (TuxBuffer_pay != NULL)
                                    bReturn = TRUE;
                        else
                        {
                                    fprintf(stderr, "tpalloc of buffer failed\n");
                                    tpterm();
                        }
                        TuxBuffer_os = (char *) tpalloc("CARRAY", NULL,
sizeof(TUX_MSG));
                        if (TuxBuffer_os != NULL)
                                    bReturn = TRUE;
                        else
                        {
                                    fprintf(stderr, "tpalloc of buffer failed\n");
                                    tpterm();
                        }
                        TuxBuffer_stock = (char *) tpalloc("CARRAY", NULL,
sizeof(TUX_MSG));
                        if (TuxBuffer_stock != NULL)
                                    bReturn = TRUE;
                        else
                        {
                                    fprintf(stderr, "tpalloc of buffer failed\n");
                                    tpterm();
                        }
*/
            }
            return bReturn;
}

/*****************************
*
*           void TuxCleanup(void)
*
*****************************/
void TuxCleanup(void)
{
            tpfree(TuxBuffer);
//          tpfree(TuxBuffer_no);
//          tpfree(TuxBuffer_pay);
//          tpfree(TuxBuffer_os);
//          tpfree(TuxBuffer_stock);
            tpterm();
}

/*****************************************************************************
*
*           BOOL TuxTransaction(char *Service, void *Data, long BufSize, long
*pnRead)
*
*****************************************************************************/
BOOL TuxTransaction(char *Service, void *Data, long BufSize, long *pnRead)
{
/*          if (strcmp(Service,"NEW_ORDER")){
                        memcpy(TuxBuffer_no, Data, BufSize);
                        TuxBuffer = TuxBuffer_no;
            }
            else if  (strcmp(Service,"PAYMENT")){
                        memcpy(TuxBuffer_pay, Data, BufSize);
                        TuxBuffer = TuxBuffer_pay;
            }
            else if  (strcmp(Service,"ORDER_STATUS")){
                        memcpy(TuxBuffer_os, Data, BufSize);
                        TuxBuffer = TuxBuffer_os;
            }
            else if (strcmp(Service,"STOCK_LEVEL")){
                        memcpy(TuxBuffer_pay, Data, BufSize);
                        TuxBuffer = TuxBuffer_stock;
            }

*/

            memcpy(TuxBuffer, Data, BufSize);
            #ifdef _DEBUG
                        fprintf(stderr,"about to tpcall Service %s,
bufsize=%d\n", Service, BufSize);

            #endif
            if (tpcall(Service, TuxBuffer, BufSize, &TuxBuffer, pnRead,
TPNOTIME) == -1)
            {
                        extern int tperrno;
                        fprintf(stderr, "TuxTransaction: tpcall failed,
tperrno=%d\n", tperrno);
                        return FALSE;
            }
            #ifdef _DEBUG
                        fprintf(stderr, "tp call retuned %d bytes\n", *pnRead);
            #endif
            if (* pnRead < BufSize)
            {
                        fprintf(stderr, "TuxTransaction: nRead(%d) <
BufSize(%d)\n", *pnRead, BufSize);
                        return FALSE;
            }
            memcpy(Data, TuxBuffer, *pnRead);
            return TRUE;
}

/*********************************************
*
*           void HandleTransactions(HANDLE hPipe)
*
*********************************************/
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;
//printf("%s", msg.Data.Trans);
                        if (! TuxTransaction(msg.Service, &msg.Data, nRead,
&nRead))
                        {
                                    fprintf(stderr, "TuxTransaction failed\n");
                                    break;
                        }
                        if (! WritePipe(hPipe, hEvent, &msg, nRead,
&nWritten))
                        {
                                    fprintf(stderr,"WritePipe Failed in
HandleTransactions()\n");
                                    break;
                        }
                        if (nWritten != nRead)
                        {
                                    fprintf(stderr, "HandleTransactions:
nWritten(%d) != nRead(%d)\n", nWritten, nRead);
                        }
            }
            CloseHandle(hEvent);
}

/*********************************************************************
*
*           BOOL StartAnother(char *name, int number, int InitialCreate)
*
*********************************************************************/
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)
*
************************************/
void Usage(char *ProgName)
{
        fprintf(stderr,"usage: %s <initial create>\n", ProgName);
}

/*******************************************
*
*       BOOL Parse(int argc, char ** argv)
*
*******************************************/
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)
*
*******************************/
void SetUpStderr(void)
{
        char buf[_MAX_PATH];
        strcpy(buf, LOG_PATH);
        _mkdir(LOG_PATH);
        sprintf(buf+ strlen(buf), LOG_NAME, ClientNumber);
        freopen(buf, "w", stderr);
        setbuf(stderr, NULL);
}

/*******************************************
*
*       int main(int argc, char ** argv)
*
*******************************************/
int main(int argc, char **argv)
{
        HANDLE hPipe;
        if (! Parse(argc, argv))
        {
                Usage(argv[ 0]);
                exit(1);
        }
        #ifdef _DEBUG
                fprintf(stderr, "client %d starting (as thread 0x%x)\n",
ClientNumber, GetCurrentThreadId());
        #endif

        SetUpStderr();

        if (!TuxInit())
        {
                fprintf(stderr, "tuxinit failed\n");
                exit(1);
        }

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

                        HandleTransactions(hPipe);

                        CloseHandle(hPipe);
        }
        TuxCleanup();
        return 0;
}
```

tux_server.c

```
#include <windows.h>
#include <stdio.h>
#include <time.h>
#include <stdarg.h>
// Tuxedo include files
#include <atmi.h>
#include <userlog.h>
// Database include files
#ifdef USE_ODBC
        #include <sqltypes.h>
        #include <sql.h>
        #include <sqlext.h>
        HENV henv;
#else
        #define DBNTWIN32
        #include <sqlfront.h>
        #include <sqldb.h>
#endif
// include files for this project
#include "util.h"
#include "trans.h"
#include "tpcc.h"
#include "sqlroutines.h"
#include "tux.h"
// Global variables
short       iMaxConnections= 1;
char        szErrorLogPath[]="\\inetpub\\wwwroot\\err_tpcc_tux.txt";
DBPROCESS *pdbproc;
char        *Server                 =NULL;
char        *Database   ="tpcc";
char        *User                   ="sa";
char        *Password   ="";
int                     spId;
TUX_DATA data;
TERM        Term;
EXTENSION_CONTROL_BLOCK *gpECB= NULL;

/************************************
*
*       void Log(char *format, ...)
*
************************************/
void Log(char *format, ...)
{
        va_list args;
        char        buf[ 4096];
        int                     len;
        va_start(args, format);
        _strtime(buf);
        strcat(buf," ");
        len = strlen(buf);
        (void)_vsnprintf(buf + len, sizeof(buf)- len- 1, format, args);
        buf[ sizeof(buf)- 1]= '\0';
        va_end(args);
        userlog(buf);
}

/*****************************************************************************
```

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

/*********************************************
*
*		BOOL IsValidTermId(int TermId)
*
*********************************************/
BOOL IsValidTermId(int TermId)
{
		return FALSE;
}

/*************************************************
*
*		int tpsvrinit(int argc, char *argv[])
*
*************************************************/
int tpsvrinit(int argc, char *argv[])
{
		char App[ 1024];
		{
			int i;
			for(i= 0; i< argc; i++)
				printf("argv[%d]=%s\n", i, argv[ i]);
		}
		Log("starting the tuxedo TPCC server");
		if (gethostname(App, sizeof(App)))
			strcpy(App, "TPCC");
		if (! SQLInit())
		{
			Log("SQLInit failed");
			return -1;
		}
		if (getenv("SERVER"))
			Server = strdup(getenv("SERVER"));
		if (Server == NULL)
		{
			Log("SERVER Environment variable not set");
			return -1;
		}

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

		return 0;
}

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

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

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

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

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

tux_sql.c

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#ifdef		USE_ODBC
		#include <sqltypes.h>
		#include <sql.h>
		#include <sqlext.h>
		HENV henv;
#else
		#define DBNTWIN32
		#include <sqlfront.h>
		#include <sqldb.h>
```

```
#endif
#include "trans.h"
#include "httpext.h"
#include "tpcc.h"
#include "tux.h"
#include "sqlroutines.h"
#include "pipe_routines.h"
#include "util.h"
const int       ARGSIZE= 1024;
const int       PIPE_BUF_SIZE= 4096;
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
*
*************************************/
BOOL SQLThreadAttach( void)
{
                THREAD_DATA                     *pData;

                #ifdef _DEBUG
                                fprintf( stderr, "SQLThread attach starts\n");
                #endif

                pData = (THREAD_DATA *) malloc( sizeof( THREAD_DATA));
                if (! pData)
                                return FALSE;
                memset( pData, 0, sizeof(* pData));

                EnterCriticalSection(&CriticalSection);
                pData->ThreadNumber = ThreadCount++;
                LeaveCriticalSection(&CriticalSection);

                pData->hPipe = OpenClientPipe( pData->ThreadNumber);
                if (pData->hPipe == INVALID_HANDLE_VALUE)
                {
                                #ifdef _DEBUG
                                                fprintf( stderr, "SQLThreadattach failed
for thread %d\n", pData->ThreadNumber);
                                #endif
                                free( pData);
                                return FALSE;
                }
                else
                                TlsSetValue( TlsIndex, pData);
                                #ifdef _DEBUG
                                                fprintf( stderr, "SQLThread attach
succeeds for thread %d\n", pData->ThreadNumber);
                                #endif
                                return TRUE;
                }

/*****************************
*
*               BOOL SQLThreadDetach
*
*****************************/
BOOL SQLThreadDetach( void)
{
                THREAD_DATA *pData = TlsGetValue( TlsIndex);
                if (pData)
                {
                                CloseHandle( pData->hPipe);
                                free( pData);
                }
                return TRUE;
}

/************************
*
*               BOOL SQLInit
*
************************/
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 SQLCleanup( void)
{
                TlsFree( TlsIndex);
                TlsIndex = 0xffffffff;
                DeleteCriticalSection(&CriticalSection);
}

/*******************************
*
*               BOOL SQLOpenConnection
*
*******************************/
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
*
*******************************/
BOOL SQLCloseConnection( EXTENSION_CONTROL_BLOCK *pECB,
DBPROCESS *dbproc)
{
                return FALSE;
}

/*****************************
*
*               BOOL TuxTransaction
*
*****************************/

BOOL TuxTransaction( char *Service, EXTENSION_CONTROL_BLOCK *pECB,
int TermId, int SyncId, DBPROCESS *dbproc, short DeadlockRetry, void *Data,
long BufSize)
{
                THREAD_DATA *pData;
                TUX_MSG                 msg;
                DWORD                   nBytes;
                PECBINFO   pECBInfo = (PECBINFO) dbproc; // forgive them
them, for they know not what they do...

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

                pData = TlsGetValue( TlsIndex);
                if (pData == NULL)
                {
                                if (! SQLThreadAttach())
                                {
                                                fprintf( stderr, "TuxTransaction: unable
to attach\n");
                                                return FALSE;
                                }
                                pData = TlsGetValue( TlsIndex);
                }

                // fill the struct to ship to tux
                strcpy( msg.Service, Service);
                msg.Data.TermId = TermId;
                msg.Data.SyncId = SyncId;
                msg.Data.DeadlockRetry = DeadlockRetry;
                msg.Data.Error = FALSE;
                memcpy(&msg.Data.Trans, Data, BufSize);
                if (! WritePipe( pData->hPipe, NULL, &msg,
MSG_HEADER_SIZE(&msg)+ BufSize, &nBytes))
                {
                                fprintf( stderr, "Tuxtransaction: WritePipe Failed\n");
                                return FALSE;
                }
                if (nBytes != MSG_HEADER_SIZE(&msg)+ BufSize)
                {
                                fprintf( stderr, "Tuxtransaction: short write, size=%d,
written=%d\n", MSG_HEADER_SIZE(&msg)+ BufSize, nBytes);
                                return FALSE;
                }
                if (! ReadPipe( pData->hPipe, NULL, &msg, sizeof( msg),
&nBytes))
                {
                                fprintf( stderr, "Tuxtransaction: ReadPipe Failed\n");
                                return FALSE;
                }
                if (msg.Data.Error)
                {
                                #ifdef _DEBUG
                                                fprintf( stderr, "msg.Error set,
ErrorMsg=%s\n", msg.Data.Trans.ErrorMsg);
                                #endif
```

```
                WriteZString( pECB, msg.Data.Trans.ErrorMsg);
        }

                // patch things up so the upper levels don't know this
went
                // through tux
                pECBInfo->iTermId = TermId;
                pECBInfo->iSyncId = SyncId;
                pECBInfo->bDeadlock = msg.Data.bDeadlock;
                pECBInfo->bFailed = msg.Data.bFailed;
                #ifdef _DEBUG
                        fprintf( stderr, "bFailed=%d\n",
pECBInfo->bFailed);
                #endif
                memcpy( Data, &msg.Data.Trans, BufSize);
                return msg.Data.Return;
}

/************************
*
*       BOOL SQLStockLevel
*
************************/

int 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
*
************************/
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
*
************************/
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
*
************************/
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)
*
****************************************/

PECBINFO SQLGetECB( PDBPROCESS p)
{
        return (PECBINFO) p;
}
```

util.c

```
#include <windows.h>
#include <string.h>
#include "util.h"
/* FUNCTION: void UtilStrCpy( char * pDest, char * pSrc, int n)
*
* PURPOSE: This function copies n characters from string pSrc to pDst and
places a
* null character at the end of the destination string.
*
* ARGUMENTS: char* pDestdestination string pointer
* char* pSrcsource string pointer
* intnnumber of characters to copy
*
* RETURNS: None
*
* COMMENTS: Unlike strncpy this function ensures that the result string is
* always null terminated.
*
*/
void UtilStrCpy( char *pDest, char *pSrc, int n)
{
strncpy( pDest, pSrc, n);
pDest[ n] = '\0';
return;
}
```

delirpt.c

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

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

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

#define INTERVAL                                        .01

        //90th percentile calculation bucket interval

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

typedef struct _RPTLINE
{
        SYSTEMTIME              start;

        //delilog report line start time
        SYSTEMTIME              end;

        //delilog report line end time
        int                                     response;

        //delilog report line time delivery took in milliseconds
        int                                     w_id;

                //delilog report line warehouse id for delivery
        int                                     o_carrier_id;

        //delilog report line carier id for delivery
        int                                     items[10];

        //delilog report line delivery line items
} RPTLINE, *PRPTLINE;

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

int                                             versionMS = 3;
                                                        //delirpt
version
int                                             versionMM = 0;
int                                             versionLS = 2;
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 ( szDelivery[0] == '*' )
                                                continue;
                                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;
                }
        }

        printf("Max Response = %d.%d
\n",iMaxSeconds/1000,iMaxSeconds%1000 );
        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 ( szDelivery[0] == '*' )
                        continue;
                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;
        }
        if ( szBuffer[0] == '*' )
        {
                //error line ignore
                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;
}
```

Appendix B : Database Design

Build Scripts

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 tpccback 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
```

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 'tpccback'
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="C:\MSSQL70_tpcc_root.
mdf",size=8MB, FILEGROWTH=0)
log on
        (name="MSSQL70_tpcc_log",filename="F:",size=50000MB,
FILEGROWTH=0)

-- create filegroups

alter database tpcc add filegroup MSSQL70_misc_fg
alter database tpcc add filegroup MSSQL70_cstock_fg

-- add files to filegroups

alter database tpcc add file
        (name="MSSQL70_misc1",filename="G:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc2",filename="H:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc3",filename="S:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc4",filename="T:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc5",filename="U:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc6",filename="V:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc7",filename="Q:",size=10000MB,
FILEGROWTH=0),
        (name="MSSQL70_misc8",filename="R:",size=10000MB,
FILEGROWTH=0)
to filegroup MSSQL70_misc_fg

alter database tpcc add file
        (name="MSSQL70_cstock1",filename="I:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock2",filename="J:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock3",filename="K:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock4",filename="L:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock5",filename="M:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock6",filename="N:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock7",filename="O:",size=11000MB,
FILEGROWTH=0),
        (name="MSSQL70_cstock8",filename="P:",size=11000MB,
FILEGROWTH=0)
to filegroup MSSQL70_cstock_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','tpccback','W:\tpccback.dmp'
go
```

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. 4.00
--          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 'history','AllowRowLocks',TRUE
go
sp_indexoption 'history','AllowPageLocks',TRUE
go
sp_indexoption 'orders','AllowRowLocks',TRUE
go
sp_indexoption 'orders','AllowPageLocks',FALSE
go
sp_indexoption 'customer','AllowRowLocks',TRUE
go
sp_indexoption 'customer','AllowPageLocks',FALSE
go
sp_indexoption 'customer','AllowRowLocks',TRUE
go
sp_indexoption 'customer','AllowPageLocks',FALSE
go
sp_indexoption 'district','AllowRowLocks',TRUE
go
sp_indexoption 'district','AllowPageLocks',FALSE
go
sp_indexoption 'warehouse','AllowRowLocks',TRUE
go
sp_indexoption 'warehouse','AllowPageLocks',FALSE
go
sp_indexoption 'stock','AllowRowLocks',TRUE
go
sp_indexoption 'stock','AllowPageLocks',FALSE
go
sp_indexoption 'order_line','AllowRowLocks',TRUE
go
sp_indexoption 'order_line','AllowPageLocks',FALSE
go
sp_indexoption 'new_order','AllowRowLocks',FALSE
go
sp_indexoption 'new_order','AllowPageLocks',TRUE
go
sp_indexoption 'item','AllowRowLocks',FALSE
```

```
go
sp_indexoption 'item','AllowPageLocks',FALSE
go


use tpcc
go

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_autostats customer,'off'
exec sp_autostats district,'off'
exec sp_autostats item,'off'
exec sp_autostats new_order,'off'
exec sp_autostats order_line,'off'
exec sp_autostats orders,'off'
exec sp_autostats stock,'off'
exec sp_autostats warehouse,'off'
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
```

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_cstock_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_cstock_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:    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
```

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_cstock_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
```

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 tpccback with stats = 1

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

go
```

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_cstock_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_cstock_fg
go
```

Stored Procedure

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


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,

                @i_id12 int = 0, @s_w_id12 smallint = 0, @ol_qty12 smallint = 0,

                @i_id13 int = 0, @s_w_id13 smallint = 0, @ol_qty13 smallint = 0,

                @i_id14 int = 0, @s_w_id14 smallint = 0, @ol_qty14 smallint = 0,

                @i_id15 int = 0, @s_w_id15 smallint = 0, @ol_qty15 smallint = 0


as
declare @w_tax          numeric(4,4),
        @d_tax          numeric(4,4),
        @c_last         char(16),
        @c_credit       char(2),
        @c_discount     numeric(4,4),
        @i_price        numeric(5,2),
        @i_name         char(24),
        @i_data         char(50),
        @o_entry_d      datetime,
        @remote_flag    int,
        @s_quantity     smallint,
        @s_data         char(50),
        @s_dist         char(24),
                @li_no          int,
                @o_id                                   int,
                @commit_flag    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,
                  @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
                        update stock
                                        set s_ytd       = s_ytd + @li_qty,
                    @s_quantity  = s_quantity,
                    s_quantity   = s_quantity - @li_qty +
                    case when (s_quantity - @li_qty < 10) then 91 else 0 end,
                    s_order_cnt  = s_order_cnt + 1,
                    s_remote_cnt = s_remote_cnt + case
                  when (@li_s_w_id = @w_id) then 0 else 1 end,
                       @s_data    = s_data,
                       @s_dist    = case @d_id
                                        when 1  then s_dist_01
                                                when 2  then s_dist_02
                                                when 3  then s_dist_03
                                                when 4  then s_dist_04
                                                when 5  then s_dist_05
                                                when 6  then s_dist_06
                                                when 7  then s_dist_07
                                                when 8  then s_dist_08
                                                when 9  then s_dist_09
                                                when 10 then s_dist_10
                                    end
            where s_i_id = @li_id and
                          s_w_id = @li_s_w_id

-- insert order_line data (using data from item and stock)

            insert into order_line values(@o_id,
                                @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
```

Loader Source Code

GETARGS.C

```
//              File:                   GETARGS.C
//                                                  Microsoft TPC-C Kit Ver.
4.00
//                                                  Copyright Microsoft, 1996,
1997, 1998
//              Purpose:    Source file for command line processing


// Includes
#include "tpcc.h"

//=========================================================
============
//
// Function name: GetArgsLoader
//
//=========================================================
============

void GetArgsLoader(int argc, char **argv, TPCCLDR_ARGS *pargs)
{
   int                  i;
   char         *ptr;

#ifdef DEBUG
   printf("[%ld]DBG: Entering GetArgsLoader()\n", (int) GetCurrentThreadId());
#endif

   /* init args struct with some useful values */
   pargs->server                        = SERVER;
   pargs->user                          = USER;
   pargs->password                      = PASSWORD;
   pargs->database                      = DATABASE;
   pargs->batch                         = BATCH;
   pargs->num_warehouses                        = UNDEF;
                pargs->tables_all                                       = TRUE;
                pargs->table_item                                       = FALSE;
                pargs->table_warehouse          = FALSE;
                pargs->table_customer           = FALSE;
                pargs->table_orders                                     = FALSE;
                pargs->loader_res_file          = LOADER_RES_FILE;
                pargs->pack_size                = DEFLDPACKSIZE;
                pargs->starting_warehouse       =
DEF_STARTING_WAREHOUSE;
                pargs->build_index                                      =
BUILD_INDEX;
                pargs->index_order                                      =
INDEX_ORDER;
                pargs->index_script_path  = INDEX_SCRIPT_PATH;
                pargs->scale_down                                       =
SCALE_DOWN;

   /* check for zero command line args */
   if ( argc == 1 )
                        GetArgsLoaderUsage();

   for (i = 1; i < argc; ++i)
   {
                        if (argv[i][0] != '-' && argv[i][0] != '/')
                        {
                        printf("\nUnrecognized command");
                        GetArgsLoaderUsage();
                        exit(1);
                        }

                        ptr = argv[i];
```

```
                switch (ptr[1])
                {
                case 'h':    /* Fall throught */
                case 'H':
                                GetArgsLoaderUsage();
                                break;

                case 'D':
                                pargs->database = ptr+2;
                                break;

                case 'P':
                                pargs->password = ptr+2;
                                break;

                case 'S':
                                pargs->server = ptr+2;
                                break;

                case 'U':
                                pargs->user = ptr+2;
                                break;

                case 'b':
                                pargs->batch = atol(ptr+2);
                                break;

                case 'W':
                                pargs->num_warehouses
= atol(ptr+2);
                                break;

                case 's':
                                pargs-
>starting_warehouse = atol(ptr+2);
                                break;

                case 't':
                                {
                                        pargs-
>tables_all = FALSE;
                                        if
(strcmp(ptr+2,"item") == 0)
                pargs->table_item = TRUE;
                                        else if
(strcmp(ptr+2,"warehouse") == 0)
                pargs->table_warehouse = TRUE;
                                        else if
(strcmp(ptr+2,"customer") == 0)
                pargs->table_customer = TRUE;
                                        else if
(strcmp(ptr+2,"orders") == 0)
                pargs->table_orders = TRUE;
                                        else
                                        {
                printf("\nUnrecognized command");
                GetArgsLoaderUsage();
                                        exit(1);
                                        }
                                        break;
                                }
                        case 'f':
                                        pargs->loader_res_file =
ptr+2;
                                        break;

                                case 'p':
                                        pargs->pack_size =
atol(ptr+2);
                                        break;

                                case 'i':
                                        pargs->build_index =
atol(ptr+2);
                                        break;

                                case 'o':
                                        pargs->index_order =
atol(ptr+2);
                                        break;

                                case 'c':
                                        pargs->scale_down =
atol(ptr+2);
                                        break;

                        case 'd':
                                        pargs->index_script_path
= ptr+2;
                                        break;

                        default:
                                GetArgsLoaderUsage();
                                exit(-1);
                                break;
                        }
        }

        /* check for required args */
        if (pargs->num_warehouses == UNDEF )
        {
                        printf("Number of Warehouses is required\n");
                        exit(-2);
        }

        return;
}


//============================================================
============
//
// Function name: GetArgsLoaderUsage
//
//============================================================
============

void GetArgsLoaderUsage()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering GetArgsLoaderUsage()\n", (int)
GetCurrentThreadId());
#endif


    printf("TPCCLDR:\n\n");
                printf("Parameter                                       Default\n");
                printf("-------------------------------------------------------------\n");
    printf("-W Number of Warehouses to Load                 Required \n");
    printf("-S Server                                   %s\n", SERVER);
    printf("-U Username                                     %s\n", USER);
    printf("-P Password                                   %s\n", PASSWORD);
    printf("-D Database                                   %s\n", DATABASE);
                printf("-b Batch Size                                   %ld\n", (long)
BATCH);
                printf("-p TDS packet size                              %ld\n", (long)
DEFLDPACKSIZE);
                printf("-f Loader Results Output Filename               %s\n",
LOADER_RES_FILE);
                printf("-s Starting Warehouse                           %ld\n", (long)
DEF_STARTING_WAREHOUSE);
                printf("-i Build Option (data = 0, data and index = 1)  %ld\n",
(long) BUILD_INDEX);
                printf("-o Cluster Index Build Order (before = 1, after = 0)  %ld\n",
(long) INDEX_ORDER);
                printf("-c Build Scaled Database (normal = 0, tiny = 1)  %ld\n",
(long) SCALE_DOWN);
                printf("-d Index Script Path                            %s\n",
INDEX_SCRIPT_PATH);
                printf("-t Table to Load                                all tables \n");
    printf("   [item|warehouse|customer|orders]\n");
    printf("   Notes: \n");
    printf("     - the '-t' parameter may be included multiple times to \n");
    printf("       specify multiple tables to be loaded \n");
    printf("     - 'item' loads ITEM table \n");
    printf("     - 'warehouse' loads WAREHOUSE, DISTRICT, and STOCK tables
\n");
    printf("     - 'customer' loads CUSTOMER and HISTORY tables \n");
    printf("     - 'orders' load NEW-ORDER, ORDERS, ORDER-LINE tables \n");

                printf("\nNote:  Command line switches are case sensitive.\n");

    exit(0);
}
```

RANDOM.C

```
//              File:                   RANDOM.C
//                                              Microsoft TPC-C Kit Ver.
4.00
//                                              Copyright Microsoft, 1996,
1997, 1998
//              Purpose:        Random number generation routines for database
loader


// Includes
#include "tpcc.h"
#include "math.h"

// Defines
#define A           16807
#define M       2147483647
#define Q           127773      /* M div A */
#define R            2836       /* M mod A */
#define Thread              __declspec(thread)

// Globals
long            Thread Seed = 0;        /* thread local seed */

/*****************************************************************************
*                                                                            *
* random -                                                                   *
*       Implements a GOOD pseudo random number generator.  This generator
*
*       will/should? run the complete period before repeating.               *
*                                                                            *
* Copied from:                                                               *
*       Random Numbers Generators: Good Ones Are Hard to Find.               *
*       Communications of the ACM - October 1988 Volume 31 Number 10         *
```

```
*                                                                       *
* Machine Dependencies:                                                 *
*       long must be 2 ^ 31 - 1 or greater.                             *
*                                                                       *
************************************************************************/

/************************************************************************
* seed - load the Seed value used in irand and drand.  Should be used before  *
*       first call to irand or drand.                                   *
************************************************************************/

void seed(long val)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering seed()...\n", (int) GetCurrentThreadId());
                printf("Old Seed %ld New Seed %ld\n",Seed, val);
#endif

    if ( val < 0 )
                val = abs(val);

    Seed = val;

}


/************************************************************************
*                                                                       *
* irand - returns a 32 bit integer pseudo random number with a period of     *
*       1 to 2 ^ 32 - 1.                                                *
*                                                                       *
* parameters:                                                           *
*       none.                                                           *
*                                                                       *
* returns:                                                              *
*       32 bit integer - defined as long ( see above ).                 *
*                                                                       *
* side effects:                                                         *
*       seed get recomputed.                                            *
************************************************************************/

long irand()
{
    register long    s;      /* copy of seed */
    register long    test;   /* test flag */
    register long    hi;     /* tmp value for speed */
    register long    lo;     /* tmp value for speed */

#ifdef DEBUG
    printf("[%ld]DBG: Entering irand()...\n", (int) GetCurrentThreadId());
#endif

    s = Seed;
    hi = s / Q;
    lo = s % Q;

    test = A * lo - R * hi;
    if ( test > 0 )
                        Seed = test;
    else
                        Seed = test + M;

    return( Seed );
}


/************************************************************************
*                                                                       *
* drand - returns a double pseudo random number between 0.0 and 1.0.        *
*       See irand.                                                      *
************************************************************************/
double drand()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering drand()...\n", (int) GetCurrentThreadId());
#endif

    return( (double)irand() / 2147483647.0);
}


//=============================================================
============
// Function   : RandomNumber
//
// Description:
//=============================================================
============
long RandomNumber(long lower, long upper)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering RandomNumber()...\n", (int) GetCurrentThreadId());
#endif

                if ( upper == lower )           /* pgd 08-13-96 perf enhancement */
                        return lower;

                upper++;

    if ( upper <= lower )
                        rand_num = upper;
    else
                        rand_num = lower + irand() % (upper - lower); /* pgd
08-13-96 perf enhancement */


#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber between %ld & %ld ==> %ld\n",
                                                        (int) GetCurrentThreadId(),
lower, upper, rand_num);
#endif


    return rand_num;
}


#if 0

//Orginal code pgd 08/13/96

long RandomNumber(long lower,
                                                        long upper)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering RandomNumber()...\n", (int) GetCurrentThreadId());
#endif

                upper++;

    if ((upper <= lower))
                        rand_num = upper;
    else
                        rand_num = lower + irand() % ((upper > lower) ?
upper - lower : upper);

#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber between %ld & %ld ==> %ld\n",
                                                        (int) GetCurrentThreadId(),
lower, upper, rand_num);
#endif


    return rand_num;
}
#endif

//=============================================================
============
// Function   : NURand
//
// Description:
//=============================================================
============
long NURand(int iConst,
                long x,
                long y,
                long C)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering NURand()...\n", (int) GetCurrentThreadId());
#endif

    rand_num = (((RandomNumber(0,iConst) | RandomNumber(x,y)) + C) % (y-
x+1))+x;

#ifdef DEBUG
    printf("[%ld]DBG: NURand: num = %d\n", (int) GetCurrentThreadId(),
rand_num);
#endif

    return rand_num;
}
```

STRINGS.C

```
//              File:                   STRINGS.C
//                                                      Microsoft TPC-C Kit Ver.
4.00
//                                                      Copyright Microsoft, 1996,
1997, 1998
//              Purpose:        Source file for database loader string functions


// Includes
#include "tpcc.h"
#include <string.h>
#include <ctype.h>

//=============================================================
============
//
// Function name: MakeAddress
//
//=============================================================
============

void MakeAddress(char *street_1,
                                                        char *street_2,
                                                        char *city,
                                                        char *state,
```

```
                                        char *zip)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAddress()\n", (int) GetCurrentThreadId());
#endif

    MakeAlphaString (10, 20, ADDRESS_LEN, street_1);
    MakeAlphaString (10, 20, ADDRESS_LEN, street_2);
    MakeAlphaString (10, 20, ADDRESS_LEN, city);
    MakeAlphaString ( 2,  2, STATE_LEN, state);
    MakeZipNumberString( 9,  9, ZIP_LEN, zip);

#ifdef DEBUG
    printf("[%ld]DBG: MakeAddress: street_1: %s, street_2: %s, city: %s, state:
%s, zip: %s\n",
                                        (int) GetCurrentThreadId(), street_1,
street_2, city, state, zip);
#endif

    return;

}


//============================================================
============
//
// Function name: LastName
//
//============================================================
============

void LastName(int  num,
                                char *name)
{
    static char *n[] =
    {
                                "BAR" , "OUGHT", "ABLE" , "PRI"  , "PRES",
                                "ESE" , "ANTI" , "CALLY", "ATION", "EING"
    };

#ifdef DEBUG
    printf("[%ld]DBG: Entering LastName()\n", (int) GetCurrentThreadId());
#endif

    if ((num >= 0) && (num < 1000))
    {
                                strcpy(name, n[(num/100)%10]);
                                strcat(name, n[(num/10)%10]);
                                strcat(name, n[(num/1)%10]);

                                if (strlen(name) < LAST_NAME_LEN)
                                {
                                        PaddString(LAST_NAME_LEN, name);
                                }

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


#ifdef DEBUG
    printf("[%ld]DBG: LastName: num = [%d] ==> [%d][%d][%d]\n",
                                        (int) GetCurrentThreadId(), num,
num/100, (num/10)%10, num%10);
                printf("[%ld]DBG: LastName: String = %s\n", (int)
GetCurrentThreadId(), name);
#endif

    return;

}


//============================================================
============
//
// Function name: MakeAlphaString
//
//============================================================
============

//philipdu 08/13/96 Changed MakeAlphaString to use A-Z, a-z, and 0-9 in
//accordance with spec see below:
//The spec says:
//4.3.2.2       The notation random a-string [x .. y]
//(respectively, n-string [x .. y]) represents a string of random alphanumeric
//(respectively, numeric) characters of a random length of minimum x, maximum
y,
//and mean (y+x)/2. Alphanumerics are A..Z, a..z, and 0..9.  The only other
//requirement is that the character set used "must be able to represent a
minimum
//of 128 different characters".  We are using 8-bit chars, so this is a non issue.
//It is completely unreasonable to stuff non-printing chars into the text fields.
//-CLevine 08/13/96

int MakeAlphaString( int  x, int  y, int  z, char *str)
{
                int                             len;
                int                             i;
                static          char chArray[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy
z";
                static          int             chArrayMax = 61;

#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeAlphaString()\n", (int) GetCurrentThreadId());
#endif

                len= RandomNumber(x, y);

                for (i=0; i<len; i++)
                                str[i] = chArray[RandomNumber(0, chArrayMax)];
                if ( len < z )
                                memset(str+len, ' ', z - len);
                str[len] = 0;

    return len;
}


//============================================================
============
//
// Function name: MakeOriginalAlphaString
//
//============================================================
============

int MakeOriginalAlphaString(int  x,

int  y,

                int  z,

char *str,
                int  percent)
{
    int                         len;
    int                         val;
    int                         start;

#ifdef DEBUG
    printf("[%ld]DBG: Entering MakeOriginalAlphaString()\n", (int)
GetCurrentThreadId());
#endif

    // verify prercentage is valid
    if ((percent < 0) || (percent > 100))
    {
                                printf("MakeOrigianlAlphaString: Invalid percentage:
%d\n", percent);
                                exit(-1);
    }

    // verify string is at least 8 chars in length
    if ((x + y) <= 8)
    {
                                printf("MakeOriginalAlphaString: string length must
be >= 8\n");
                                exit(-1);
    }

    // Make Alpha String
    len = MakeAlphaString(x,y, z, str);

    val = RandomNumber(1,100);
    if (val <= percent)
    {
                                start = RandomNumber(0, len - 8);
                                strncpy(str + start, "ORIGINAL", 8);
    }


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

    return strlen(str);
}


//============================================================
============
//
// Function name: MakeNumberString
//
//============================================================
============
int MakeNumberString(int  x, int  y, int  z, char *str)
{
                char tmp[16];

                //MakeNumberString is always called MakeZipNumberString(16,
16, 16, string)

                memset(str, '0', 16);
                itoa(RandomNumber(0, 99999999), tmp, 10);
                memcpy(str, tmp, strlen(tmp));

                itoa(RandomNumber(0, 99999999), tmp, 10);
                memcpy(str+8, tmp, strlen(tmp));

                str[16] = 0;
```

```
    return 16;
}


//=============================================================
============
//
// Function name: MakeZipNumberString
//
//=============================================================
============
int MakeZipNumberString(int  x, int  y, int  z, char *str)
{
        char tmp[16];

        //MakeZipNumberString is always called MakeZipNumberString(9,
9, 9, string)

        strcpy(str, "000011111");

        itoa(RandomNumber(0, 9999), tmp, 10);
        memcpy(str, tmp, strlen(tmp));

    return 9;
}


//=============================================================
============
//
// Function name: InitString
//
//=============================================================
============
void InitString(char *str, int len)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering InitString()\n", (int) GetCurrentThreadId());
#endif

        memset(str, ' ', len);
    str[len] = 0;
}


//=============================================================
============
// Function name: InitAddress
//
// Description:
//
//=============================================================
============

void InitAddress(char *street_1, char *street_2, char *city, char *state, char *zip)
{
        memset(street_1, ' ', ADDRESS_LEN+1);
        memset(street_2, ' ', ADDRESS_LEN+1);
        memset(city, ' ', ADDRESS_LEN+1);

    street_1[ADDRESS_LEN+1] = 0;
    street_2[ADDRESS_LEN+1] = 0;
    city[ADDRESS_LEN+1]     = 0;

        memset(state, ' ', STATE_LEN+1);
    state[STATE_LEN+1] = 0;

        memset(zip, ' ', ZIP_LEN+1);
    zip[ZIP_LEN+1] = 0;
}
```

```
//=============================================================
============
//
// Function name: PaddString
//
//=============================================================
============

void PaddString(int max, char *name)
{
        int                     len;

        len = strlen(name);
        if ( len < max )
                memset(name+len, ' ', max - len);
        name[max] = 0;

        return;
}
```

TIME.C

```
//      File:           TIME.C
//                                              Microsoft TPC-C Kit Ver.
4.00
//                                              Copyright Microsoft, 1996,
1997, 1998
//      Purpose:    Source file for time functions


// Includes
#include "tpcc.h"

// Globals
static long start_sec;


//=============================================================
============
//
// Function name: TimeNow
//
//=============================================================
============

long TimeNow()
{
  long                  time_now;
  struct        _timeb el_time;

#ifdef DEBUG
    printf("[%ld]DBG: Entering TimeNow()\n", (int) GetCurrentThreadId());
#endif

  _ftime(&el_time);

  time_now = ((el_time.time - start_sec) * 1000) + el_time.millitm;

  return time_now;
}
```

TPCC.H

```
//      File:           TPCC.H
//                                              Microsoft TPC-C Kit Ver.
4.00
//                                              Copyright Microsoft,
1996, 1997, 1998
//      Purpose:    Header file for TPC-C database loader


// Build number of TPC Benchmark Kit
#define TPCKIT_VER      "4.00"

// General headers
#include <windows.h>
#include <winbase.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>
#include <stddef.h>
#include <stdarg.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <sys\types.h>

// ODBC headers
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>

// General constants
#define MILLI                   1000
#define FALSE                   0
#define TRUE                    1
#define UNDEF
-1
#define MINPRINTASCII           32
#define MAXPRINTASCII           126

// Default environment constants
#define SERVER                  ""
#define DATABASE                "tpcc"
#define USER                    "sa"
#define PASSWORD                ""

// Default loader arguments
#define BATCH                   10000
#define DEFLDPACKSIZE                   32768
#define ORDERS_PER_DIST         3000
#define LOADER_RES_FILE         "logs\\load.out"
#define LOADER_NURAND_C                 123
#define DEF_STARTING_WAREHOUSE          1
#define BUILD_INDEX
        1       // build both data and indexes
#define INDEX_ORDER
        1       // build indexes before load
#define SCALE_DOWN
        0   // build a normal scale database
#define INDEX_SCRIPT_PATH       "scripts"

typedef struct
{
   char                         *server;
   char                         *database;
   char                         *user;
   char                         *password;
        BOOL                            tables_all;
                // set if loading all tables
```

```
        BOOL                                    table_item;
                // set if loading ITEM table specifically
        BOOL
        table_warehouse;        // set if loading WAREHOUSE,
DISTRICT, and STOCK
        BOOL
        table_customer;                 // set if loading
CUSTOMER and HISTORY
        BOOL
        table_orders;                   // set if loading NEW-
ORDER, ORDERS, ORDER-LINE
   long                                 num_warehouses;
   long                                 batch;
   long                                 verbose;
        long                                    pack_size;
        char
        *loader_res_file;
        char
        *synch_servername;
        long
        case_sensitivity;
        long
        starting_warehouse;
        long                                    build_index;
        long                                    index_order;
        long                                    scale_down;
        char
        *index_script_path;
} TPCCLDR_ARGS;

// 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 CREDIT_LEN              2
        #define C_DATA_LEN              500
        #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
        #define C_SINCE_LEN
23
        #define H_DATE_LEN
23
        #define OL_DELIVERY_D_LEN                       23
        #define O_ENTRY_D_LEN           23


// Functions in random.c
void        seed();
long        irand();
double      drand();
void        WUCreate();
short       WURand();
long        RandomNumber(long lower, long upper);

// Functions in getargs.c;
void        GetArgsLoader();
void        GetArgsLoaderUsage();

// Functions in time.c
long        TimeNow();

// Functions in strings.c
void        MakeAddress();
void        LastName();
int         MakeAlphaString();
int         MakeOriginalAlphaString();
int         MakeNumberString();
int         MakeZipNumberString();
void        InitString();
void        InitAddress();
void        PaddString();
```

TPCCLDR.C

```
//          File:                   TPCCLDR.C
//                                              Microsoft TPC-C Kit Ver.
4.00
//                                              Copyright Microsoft, 1996,
1997, 1998
//          Purpose:    Source file for TPC-C database loader


// Includes
#include "tpcc.h"
#include "search.h"

// Defines
#define MAXITEMS            100000
#define MAXITEMS_SCALE_DOWN                     100
#define CUSTOMERS_PER_DISTRICT  3000
#define CUSTOMERS_SCALE_DOWN 30
#define DISTRICT_PER_WAREHOUSE  10
#define ORDERS_PER_DISTRICT    3000
#define ORDERS_SCALE_DOWN      30
#define MAX_CUSTOMER_THREADS   2
#define MAX_ORDER_THREADS      3
#define MAX_MAIN_THREADS       4

// Functions declarations

void HandleErrorDBC (SQLHDBC  hdbc1);

long NURand();
void LoadItem();
void LoadWarehouse();

void Stock();
void District();

void LoadCustomer();
void CustomerBufInit();
void CustomerBufLoad();
void LoadCustomerTable();
void LoadHistoryTable();
void LoadOrders();
void OrdersBufInit();
void OrdersBufLoad();
void LoadOrdersTable();
void LoadNewOrderTable();
void LoadOrderLineTable();
void GetPermutation();
void CheckForCommit();
void OpenConnections();
void BuildIndex();
void FormatDate ();

// Shared memory structures

typedef struct
{
   long         ol;
   long         ol_i_id;
   short        ol_supply_w_id;
   short        ol_quantity;
   double        ol_amount;
   char         ol_dist_info[DIST_INFO_LEN+1];
            char
            ol_delivery_d[OL_DELIVERY_D_LEN+1];
} ORDER_LINE_STRUCT;

typedef struct
{
   long         o_id;
   short        o_d_id;
   short        o_w_id;
   long         o_c_id;
   short        o_carrier_id;
   short        o_ol_cnt;
   short        o_all_local;
            ORDER_LINE_STRUCT   o_ol[15];
} ORDERS_STRUCT;

typedef struct
{
   long                                         c_id;
   short                                        c_d_id;
   short                                        c_w_id;
            char
            c_first[FIRST_NAME_LEN+1];
   char
            c_middle[MIDDLE_NAME_LEN+1];
   char
            c_last[LAST_NAME_LEN+1];
   char
            c_street_1[ADDRESS_LEN+1];
   char
            c_street_2[ADDRESS_LEN+1];
   char                                         c_city[ADDRESS_LEN+1];
   char                                         c_state[STATE_LEN+1];
   char                                         c_zip[ZIP_LEN+1];
   char                                         c_phone[PHONE_LEN+1];
   char                                         c_credit[CREDIT_LEN+1];
   double                                       c_credit_lim;
   double                                       c_discount;
//  fix to avoid ODBC float to numeric conversion problem.
//  double                                       c_balance;
            char
            c_balance[6];

   double                                       c_ytd_payment;
            short
            c_payment_cnt;
            short
            c_delivery_cnt;
```

```
        char
        c_data[C_DATA_LEN+1];
        double                                  h_amount;
    char                                        h_data[H_DATA_LEN+1];
} CUSTOMER_STRUCT;

typedef struct
{
        char
        c_last[LAST_NAME_LEN+1];
        char
        c_first[FIRST_NAME_LEN+1];
    long                                        c_id;
} CUSTOMER_SORT_STRUCT;

typedef struct
{
    long        time_start;
} LOADER_TIME_STRUCT;


// Global variables

char        szLastError[300];

HENV        henv;

HDBC        i_hdbc1;
            // for ITEM table
HDBC        w_hdbc1;
            // for WAREHOUSE, DISTRICT, STOCK
HDBC        c_hdbc1;
            // for CUSTOMER
HDBC        c_hdbc2;
            // for HISTORY
HDBC        o_hdbc1;
            // for ORDERS
HDBC        o_hdbc2;
            // for NEW-ORDER
HDBC        o_hdbc3;
            // for ORDER-LINE

HSTMT       i_hstmt1;
HSTMT       w_hstmt1;
HSTMT       c_hstmt1, c_hstmt2;
HSTMT       o_hstmt1, o_hstmt2, o_hstmt3;

ORDERS_STRUCT   orders_buf[ORDERS_PER_DISTRICT];
CUSTOMER_STRUCT customer_buf[CUSTOMERS_PER_DISTRICT];
long        orders_rows_loaded;
long        new_order_rows_loaded;
long        order_line_rows_loaded;
long        history_rows_loaded;
long        customer_rows_loaded;
long        stock_rows_loaded;
long        district_rows_loaded;
long        item_rows_loaded;
long        warehouse_rows_loaded;
long        main_time_start;
long        main_time_end;
long                                    max_items;
long                                    customers_per_district;
long                                    orders_per_district;
long                                    first_new_order;
long                                    last_new_order;


TPCCLDR_ARGS    *aptr, args;
```

```
//============================================================
============
//
// Function name: main
//
//============================================================
============

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

        DWORD       dwThreadID[MAX_MAIN_THREADS];
        HANDLE      hThread[MAX_MAIN_THREADS];
        FILE        *fLoader;
        char        buffer[255];
        int                                             i;

        for (i=0; i<MAX_MAIN_THREADS; i++)
                hThread[i] = NULL;

    printf("\n*********************************************************");
        printf("\n*                                                   *");
        printf("\n*  Microsoft SQL Server                             *");
        printf("\n*                                                   *");
        printf("\n*  TPC-C BENCHMARK KIT:  Database loader            *");
        printf("\n*  Version %s                                     *", TPCKIT_VER);
        printf("\n*                                                   *");
        printf("\n*********************************************************\n\n");

    // process command line arguments

    aptr = &args;
    GetArgsLoader(argc, argv, aptr);

        printf("Build interface is ODBC.\n");

        if (aptr->build_index == 0)
                printf("Data load only - no index creation.\n");
        else
                printf("Data load and index creation.\n");

        if (aptr->index_order == 0)
                printf("Clustered indexes will be created after bulk
load.\n");
        else
                printf("Clustered indexes will be created before bulk
load.\n");

        // set database scale values
        if (aptr->scale_down == 1)
        {
                printf("*** Scaled Down Database ***\n");
                max_items = MAXITEMS_SCALE_DOWN;
                customers_per_district =
CUSTOMERS_SCALE_DOWN;
                orders_per_district = ORDERS_SCALE_DOWN;
                first_new_order = 0;
                last_new_order  = 30;
        }
        else
        {
                max_items = MAXITEMS;
                customers_per_district =
CUSTOMERS_PER_DISTRICT;
                orders_per_district = ORDERS_PER_DISTRICT;
                first_new_order = 2100;
                last_new_order  = 3000;
        }

        // open connections to SQL Server
        OpenConnections();

        // open file for loader results
    fLoader = fopen(aptr->loader_res_file, "w");

    if (fLoader == NULL)
        {
                printf("Error, loader result file open failed.");
                exit(-1);
        }

    // start loading data

    sprintf(buffer,"TPC-C load started for %ld warehouses.\n",aptr-
>num_warehouses);

        printf("%s",buffer);
        fprintf(fLoader,"%s",buffer);

        main_time_start = (TimeNow() / MILLI);

        // start parallel load threads


        if (aptr->tables_all || aptr->table_item)
        {
                fprintf(fLoader, "\nStarting loader threads for: item\n");

                hThread[0] = CreateThread(NULL,

                  0,

                  (LPTHREAD_START_ROUTINE) LoadItem,

                  NULL,

                  0,

                  &dwThreadID[0]);

                if (hThread[0] == NULL)
                {
                        printf("Error, failed in creating creating
thread = 0.\n");
                        exit(-1);
                }
        }

        if (aptr->tables_all || aptr->table_warehouse)
        {
                fprintf(fLoader, "Starting loader threads for:
warehouse\n");

                hThread[1] = CreateThread(NULL,

                  0,

                  (LPTHREAD_START_ROUTINE) LoadWarehouse,

                  NULL,

                  0,

                  &dwThreadID[1]);

                if (hThread[1] == NULL)
                {
                        printf("Error, failed in creating creating
thread = 1.\n");
                        exit(-1);
                }
```

```
        }

        if (aptr->tables_all || aptr->table_customer)
        {
                fprintf(fLoader, "Starting loader threads for:
customer\n");

                hThread[2] = CreateThread(NULL,

                  0,

                  (LPTHREAD_START_ROUTINE) LoadCustomer,

                  NULL,

                  0,

                  &dwThreadID[2]);

                if (hThread[2] == NULL)
                {
                        printf("Error, failed in creating creating
main thread = 2.\n");
                        exit(-1);
                }
        }

        if (aptr->tables_all || aptr->table_orders)
        {
                fprintf(fLoader, "Starting loader threads for: orders\n");

                hThread[3] = CreateThread(NULL,

                  0,

                  (LPTHREAD_START_ROUTINE) LoadOrders,

                  NULL,

                  0,

                  &dwThreadID[3]);

                if (hThread[3] == NULL)
                {
                        printf("Error, failed in creating creating
main thread = 3.\n");
                        exit(-1);
                }
        }

        // Wait for threads to finish...
        for (i=0; i<MAX_MAIN_THREADS; i++)
        {
                if (hThread[i] != NULL)
                {
                        WaitForSingleObject( hThread[i],
INFINITE );
                        CloseHandle(hThread[i]);
                        hThread[i] = NULL;
                }
        }

   main_time_end = (TimeNow() / MILLI);

   sprintf(buffer,"\nTPC-C load completed successfully in %ld minutes.\n",
                        (main_time_end - main_time_start)/60);

        printf("%s",buffer);
        fprintf(fLoader, "%s", buffer);

        fclose(fLoader);

        SQLFreeEnv(henv);

   exit(0);

        return 0;
}


//============================================================
============
//
// Function name: LoadItem
//
//============================================================
============

void LoadItem()
{
   long             i_id;
        long                i_im_id;
   char             i_name[I_NAME_LEN+1];
   double           i_price;
   char             i_data[I_DATA_LEN+1];
        char                name[20];
        long                time_start;
        RETCODE             rc;
        DBINT               rcint;
        char                bcphint[128];

        // Seed with unique number
        seed(1);

        printf("Loading item table...\n");

        // if build index before load
        if ((aptr->build_index == 1) && (aptr->index_order == 1))
                BuildIndex("idxitmcl");

        InitString(i_name, I_NAME_LEN+1);
        InitString(i_data, I_DATA_LEN+1);

        sprintf(name, "%s..%s", aptr->database, "item");

        rc = bcp_init(i_hdbc1, name, NULL, "logs\\item.err", DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        if ((aptr->build_index == 1) && (aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock, order (i_id),
ROWS_PER_BATCH = 100000");
                rc = bcp_control(i_hdbc1, BCPHINTS, (void *)
bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(i_hdbc1);
        }

        rc = bcp_bind(i_hdbc1, (BYTE *) &i_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) &i_im_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) i_name, 0, I_NAME_LEN, NULL,
0, 0, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) &i_price, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 4);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        rc = bcp_bind(i_hdbc1, (BYTE *) i_data, 0, I_DATA_LEN, NULL, 0,
0, 5);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        time_start = (TimeNow() / MILLI);

        item_rows_loaded = 0;

        for (i_id = 1; i_id <= max_items; i_id++)
        {
                i_im_id = RandomNumber(1L, 10000L);

                MakeAlphaString(14, 24, I_NAME_LEN, i_name);

                i_price = ((float) RandomNumber(100L,
10000L))/100.0;

                MakeOriginalAlphaString(26, 50, I_DATA_LEN,
i_data, 10);

                rc = bcp_sendrow(i_hdbc1);

                if (rc != SUCCEED)
                        HandleErrorDBC(i_hdbc1);

                item_rows_loaded++;
                CheckForCommit(i_hdbc1, i_hstmt1,
item_rows_loaded, "item", &time_start);
        }

        rcint = bcp_done(i_hdbc1);
        if (rcint < 0)
                HandleErrorDBC(i_hdbc1);

        printf("Finished loading item table.\n");

        SQLFreeStmt(i_hstmt1, SQL_DROP);
        SQLDisconnect(i_hdbc1);
        SQLFreeConnect(i_hdbc1);

        // if build index after load
        if ((aptr->build_index == 1) && (aptr->index_order == 0))
                BuildIndex("idxitmcl");

}


//============================================================
==================
//
// Function   : LoadWarehouse
//
// Loads WAREHOUSE table and loads Stock and District as Warehouses are
created
//
//============================================================
==================

void LoadWarehouse()
{
   short        w_id;
   char         w_name[W_NAME_LEN+1];
```

```
    char        w_street_1[ADDRESS_LEN+1];
    char        w_street_2[ADDRESS_LEN+1];
    char        w_city[ADDRESS_LEN+1];
    char        w_state[STATE_LEN+1];
    char        w_zip[ZIP_LEN+1];
    double      w_tax;
    double      w_ytd;
                char            name[20];
                long            time_start;
                RETCODE         rc;
                DBINT           rcint;
                char            bcphint[128];


                // Seed with unique number
                seed(2);

                printf("Loading warehouse table...\n");

                // if build index before load...
                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                        BuildIndex("idxwarcl");

                InitString(w_name, W_NAME_LEN+1);
                InitAddress(w_street_1, w_street_2, w_city, w_state, w_zip);

                sprintf(name, "%s..%s", aptr->database, "warehouse");

                rc = bcp_init(w_hdbc1, name, NULL, "logs\\whouse.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                        sprintf(bcphint, "tablock, order (w_id),
ROWS_PER_BATCH = %d", aptr->num_warehouses);
                        rc = bcp_control(w_hdbc1, BCPHINTS, (void *)
bcphint);
                        if (rc != SUCCEED)
                                HandleErrorDBC(w_hdbc1);
                }

                rc = bcp_bind(w_hdbc1, (BYTE *) &w_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 1);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) w_name, 0, W_NAME_LEN,
NULL, 0, 0, 2);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) w_street_1, 0, ADDRESS_LEN,
NULL, 0, 0, 3);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) w_street_2, 0, ADDRESS_LEN,
NULL, 0, 0, 4);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) w_city, 0, ADDRESS_LEN,
NULL, 0, 0, 5);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) w_state, 0, STATE_LEN, NULL,
0, 0, 6);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) w_zip, 0, ZIP_LEN, NULL, 0, 0,
7);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) &w_tax, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 8);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) &w_ytd, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 9);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                time_start = (TimeNow() / MILLI);

                warehouse_rows_loaded = 0;

                for (w_id = (short)aptr->starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
                {
                        MakeAlphaString(6,10, W_NAME_LEN, w_name);

                        MakeAddress(w_street_1, w_street_2, w_city,
w_state, w_zip);

                        w_tax = ((float)
RandomNumber(0L,2000L))/10000.00;

                        w_ytd = 300000.00;

                        rc = bcp_sendrow(w_hdbc1);
                        if (rc != SUCCEED)
                                HandleErrorDBC(w_hdbc1);

                        warehouse_rows_loaded++;
                        CheckForCommit(w_hdbc1, i_hstmt1,
warehouse_rows_loaded, "warehouse", &time_start);
                }

                rcint = bcp_done(w_hdbc1);
                if (rcint < 0)
                        HandleErrorDBC(w_hdbc1);

                printf("Finished loading warehouse table.\n");

                // if build index after load...
                if ((aptr->build_index == 1) && (aptr->index_order == 0))
                        BuildIndex("idxwarcl");

                stock_rows_loaded = 0;
                district_rows_loaded = 0;

                District();
                Stock();

}


//==============================================================
============
//
// Function   : District
//
//==============================================================
============

void District()
{
    short       d_id;
    short       d_w_id;
    char        d_name[D_NAME_LEN+1];
    char        d_street_1[ADDRESS_LEN+1];
    char        d_street_2[ADDRESS_LEN+1];
    char        d_city[ADDRESS_LEN+1];
    char        d_state[STATE_LEN+1];
    char        d_zip[ZIP_LEN+1];
    double      d_tax;
    double      d_ytd;
                char            name[20];
    long        d_next_o_id;
                long            time_start;
                int                             w_id;
                RETCODE         rc;
                DBINT           rcint;
                char            bcphint[128];

                // Seed with unique number
                seed(4);

                printf("Loading district table...\n");

                // build index before load
                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                        BuildIndex("idxdiscl");

                InitString(d_name, D_NAME_LEN+1);
                InitAddress(d_street_1, d_street_2, d_city, d_state, d_zip);
                sprintf(name, "%s..%s", aptr->database, "district");

                rc = bcp_init(w_hdbc1, name, NULL, "logs\\district.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                        sprintf(bcphint, "tablock, order (d_w_id, d_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 10));
                        rc = bcp_control(w_hdbc1, BCPHINTS, (void *)
bcphint);
                        if (rc != SUCCEED)
                                HandleErrorDBC(w_hdbc1);
                }

                rc = bcp_bind(w_hdbc1, (BYTE *) &d_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 1);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) &d_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, 2);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) d_name, 0, D_NAME_LEN,
NULL, 0, 0, 3);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) d_street_1, 0, ADDRESS_LEN,
NULL, 0, 0, 4);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) d_street_2, 0, ADDRESS_LEN,
NULL, 0, 0, 5);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);
```

```
		rc = bcp_bind(w_hdbc1, (BYTE *) d_city, 0, ADDRESS_LEN,
NULL, 0, 0, 6);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) d_state, 0, STATE_LEN, NULL,
0, 0, 7);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) d_zip, 0, ZIP_LEN, NULL, 0, 0,
8);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &d_tax, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 9);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &d_ytd, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, 10);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &d_next_o_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, 11);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		d_ytd  = 30000.0;

		d_next_o_id = orders_per_district+1;

		time_start = (TimeNow() / MILLI);

		for (w_id = aptr->starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
		{
			d_w_id = w_id;

			for (d_id = 1; d_id <=
DISTRICT_PER_WAREHOUSE; d_id++)
			{
				MakeAlphaString(6,10,D_NAME_LEN,
d_name);

				MakeAddress(d_street_1, d_street_2,
d_city, d_state, d_zip);

				d_tax = ((float)
RandomNumber(0L,2000L))/10000.00;

				rc = bcp_sendrow(w_hdbc1);
				if (rc != SUCCEED)

		HandleErrorDBC(w_hdbc1);

				district_rows_loaded++;
				CheckForCommit(w_hdbc1, w_hstmt1,
district_rows_loaded, "district", &time_start);
			}
	}

		rcint = bcp_done(w_hdbc1);
		if (rcint < 0)
			HandleErrorDBC(w_hdbc1);

		printf("Finished loading district table.\n");

		// if build index after load...
		if ((aptr->build_index == 1) && (aptr->index_order == 0))
			BuildIndex("idxdiscl");

	return;
}


//=============================================================
============
//
// Function   : Stock
//
//=============================================================
============

void Stock()
{
	long		s_i_id;
	short		s_w_id;
	short		s_quantity;
	char		s_dist_01[S_DIST_LEN+1];
	char		s_dist_02[S_DIST_LEN+1];
	char		s_dist_03[S_DIST_LEN+1];
	char		s_dist_04[S_DIST_LEN+1];
	char		s_dist_05[S_DIST_LEN+1];
	char		s_dist_06[S_DIST_LEN+1];
	char		s_dist_07[S_DIST_LEN+1];
	char		s_dist_08[S_DIST_LEN+1];
	char		s_dist_09[S_DIST_LEN+1];
	char		s_dist_10[S_DIST_LEN+1];
	long		s_ytd;
	short		s_order_cnt;
	short		s_remote_cnt;
	char		s_data[S_DATA_LEN+1];
	short		len;
		char		name[20];
		long		time_start;
		RETCODE	rc;
		DBINT		rcint;
		char		bcphint[128];

		// Seed with unique number
		seed(3);

		// if build index before load...
		if ((aptr->build_index == 1) && (aptr->index_order == 1))
			BuildIndex("idxstkcl");

		sprintf(name, "%s..%s", aptr->database, "stock");

		rc = bcp_init(w_hdbc1, name, NULL, "logs\\stock.err", DB_IN);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		if ((aptr->build_index == 1) && (aptr->index_order == 1))
		{
			sprintf(bcphint, "tablock, order (s_i_id, s_w_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 100000));
			rc = bcp_control(w_hdbc1, BCPHINTS, (void *)
bcphint);
			if (rc != SUCCEED)
				HandleErrorDBC(w_hdbc1);
		}

		rc = bcp_bind(w_hdbc1, (BYTE *) &s_i_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, 1);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		bcp_bind(w_hdbc1, (BYTE *) &s_w_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 2);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &s_quantity, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, 3);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_01, 0, S_DIST_LEN,
NULL, 0, 0, 4);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_02, 0, S_DIST_LEN,
NULL, 0, 0, 5);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_03, 0, S_DIST_LEN,
NULL, 0, 0, 6);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_04, 0, S_DIST_LEN,
NULL, 0, 0, 7);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_05, 0, S_DIST_LEN,
NULL, 0, 0, 8);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_06, 0, S_DIST_LEN,
NULL, 0, 0, 9);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_07, 0, S_DIST_LEN,
NULL, 0, 0, 10);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_08, 0, S_DIST_LEN,
NULL, 0, 0, 11);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_09, 0, S_DIST_LEN,
NULL, 0, 0, 12);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_10, 0, S_DIST_LEN,
NULL, 0, 0, 13);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &s_ytd, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, 14);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &s_order_cnt, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, 15);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = bcp_bind(w_hdbc1, (BYTE *) &s_remote_cnt, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, 16);
		if (rc != SUCCEED)
```

```
                        HandleErrorDBC(w_hdbc1);

                rc = bcp_bind(w_hdbc1, (BYTE *) s_data, 0, S_DATA_LEN, NULL,
0, 0, 17);
                if (rc != SUCCEED)
                        HandleErrorDBC(w_hdbc1);

                s_ytd = s_order_cnt = s_remote_cnt = 0;

                time_start = (TimeNow() / MILLI);

                printf("...Loading stock table\n");

                for (s_i_id=1; s_i_id <= max_items; s_i_id++)
                {

                        for (s_w_id = (short)aptr->starting_warehouse;
s_w_id <= aptr->num_warehouses; s_w_id++)
                        {

                                s_quantity =
(short)RandomNumber(10L,100L);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_01);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_02);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_03);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_04);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_05);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_06);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_07);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_08);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_09);
                                len =
MakeAlphaString(24,24,S_DIST_LEN, s_dist_10);

                                len = MakeOriginalAlphaString(26,50,
S_DATA_LEN, s_data,10);

                                rc = bcp_sendrow(w_hdbc1);
                                if (rc != SUCCEED)

                HandleErrorDBC(w_hdbc1);

                                stock_rows_loaded++;
                                CheckForCommit(w_hdbc1, w_hstmt1,
stock_rows_loaded, "stock", &time_start);

                        }
                }

                rcint = bcp_done(w_hdbc1);
                if (rcint < 0)
                        HandleErrorDBC(w_hdbc1);

                printf("Finished loading stock table.\n");

                SQLFreeStmt(w_hstmt1, SQL_DROP);
                SQLDisconnect(w_hdbc1);
                SQLFreeConnect(w_hdbc1);

                // if build index after load...
                if ((aptr->build_index == 1) && (aptr->index_order == 0))
                        BuildIndex("idxstkcl");

                return;
}


//=============================================================
============
//
// Function   : LoadCustomer
//
//=============================================================
============

void LoadCustomer()
{
                LOADER_TIME_STRUCT      customer_time_start;
                LOADER_TIME_STRUCT      history_time_start;
                short                   w_id;
   short                d_id;
                DWORD
                dwThreadID[MAX_CUSTOMER_THREADS];
                HANDLE
                hThread[MAX_CUSTOMER_THREADS];
                char                    name[20];
                char
                sqlcmd[30];
                RETCODE
                rc;
                DBINT
                rcint;
                char
                bcphint[128];

                // Seed with unique number
                seed(5);

                printf("Loading customer and history tables...\n");

                // if build index before load...
                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                        BuildIndex("idxcuscl");

                // Initialize bulk copy
                sprintf(name, "%s..%s", aptr->database, "customer");

                rc = bcp_init(c_hdbc1, name, NULL, "logs\\customer.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc1);

                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                        sprintf(bcphint, "tablock, order (c_w_id, c_d_id, c_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 30000));
                        rc = bcp_control(c_hdbc1, BCPHINTS, (void *)
bcphint);
                        if (rc != SUCCEED)
                                HandleErrorDBC(c_hdbc1);
                }

                sprintf(name, "%s..%s", aptr->database, "history");

                rc = bcp_init(c_hdbc2, name, NULL, "logs\\history.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

                sprintf(bcphint, "tablock");
                rc = bcp_control(c_hdbc2, BCPHINTS, (void *) bcphint);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

                customer_rows_loaded    = 0;
                history_rows_loaded     = 0;

                CustomerBufInit();

                customer_time_start.time_start = (TimeNow() / MILLI);
                history_time_start.time_start = (TimeNow() / MILLI);

                for (w_id = (short)aptr->starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
                {
                        for (d_id = 1; d_id <=
DISTRICT_PER_WAREHOUSE; d_id++)
                        {

                                CustomerBufLoad(d_id, w_id);

                                // Start parallel loading threads here...

                                // Start customer table thread

                                printf("...Loading customer table for:
d_id = %d, w_id = %d\n", d_id, w_id);

                                hThread[0] = CreateThread(NULL,

                                 0,

                                 (LPTHREAD_START_ROUTINE)
LoadCustomerTable,

                                 &customer_time_start,

                                 0,

                                 &dwThreadID[0]);

                                if (hThread[0] == NULL)
                                {
                                        printf("Error, failed in
creating creating thread = 0.\n");
                                        exit(-1);
                                }

                                // Start History table thread

                                printf("...Loading history table for: d_id =
%d, w_id = %d\n", d_id, w_id);

                                hThread[1] = CreateThread(NULL,

                                 0,

                                 (LPTHREAD_START_ROUTINE)
LoadHistoryTable,

                                 &history_time_start,

                                 0,

                                 &dwThreadID[1]);

                                if (hThread[1] == NULL)
                                {
                                        printf("Error, failed in
creating creating thread = 1.\n");
                                        exit(-1);
                                }

                                WaitForSingleObject( hThread[0],
INFINITE );
```

```
                                        WaitForSingleObject( hThread[1],
INFINITE );

                                        if (CloseHandle(hThread[0]) == FALSE)
                                        {
                                                printf("Error, failed in
closing customer thread handle with errno: %d\n", GetLastError());
                                        }

                                        if (CloseHandle(hThread[1]) == FALSE)
                                        {
                                                printf("Error, failed in
closing history thread handle with errno: %d\n", GetLastError());
                                        }


                        }

                }

                // flush the bulk connection
                rcint = bcp_done(c_hdbc1);
                if (rcint < 0)
                        HandleErrorDBC(c_hdbc1);

                rcint = bcp_done(c_hdbc2);
                if (rcint < 0)
                        HandleErrorDBC(c_hdbc2);

                printf("Finished loading customer table.\n");

                // if build index after load...
                if ((aptr->build_index == 1) && (aptr->index_order == 0))
                        BuildIndex("idxcuscl");

                // build non-clustered index
                if (aptr->build_index == 1)
                        BuildIndex("idxcusnc");

                // Modification added 7/21/98
                //
                // Updates Customer (C_ID = 1, C_W_ID = 1, and C_D_ID = 1) to
include
                // the LOADER_NURAND_C in the C_FIRST column
                //

                sprintf(sqlcmd,"update customer set c_first = 'C_LOAD = %d'
where c_w_id = 1 and c_d_id = 1 and c_id = 1", LOADER_NURAND_C);
                rc = SQLExecDirect(c_hdbc1, sqlcmd, SQL_NTS);

                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc1);

                SQLFreeStmt(c_hstmt1, SQL_DROP);
                SQLDisconnect(c_hdbc1);
                SQLFreeConnect(c_hdbc1);

                SQLFreeStmt(c_hstmt2, SQL_DROP);
                SQLDisconnect(c_hdbc2);
                SQLFreeConnect(c_hdbc2);


    return;

}


//============================================================
============
//
// Function   : CustomerBufInit
//
//============================================================
============

void CustomerBufInit()
{
                int     i;

                for (i=0;i<customers_per_district;i++)
                {
                        customer_buf[i].c_id = 0;
                        customer_buf[i].c_d_id = 0;
                        customer_buf[i].c_w_id = 0;

                        strcpy(customer_buf[i].c_first,"");
                        strcpy(customer_buf[i].c_middle,"");
                        strcpy(customer_buf[i].c_last,"");
                        strcpy(customer_buf[i].c_street_1,"");
                        strcpy(customer_buf[i].c_street_2,"");
                        strcpy(customer_buf[i].c_city,"");
                        strcpy(customer_buf[i].c_state,"");
                        strcpy(customer_buf[i].c_zip,"");
                        strcpy(customer_buf[i].c_phone,"");
                        strcpy(customer_buf[i].c_credit,"");

                        customer_buf[i].c_credit_lim = 0;
                        customer_buf[i].c_discount = (float) 0;

                        //  fix to avoid ODBC float to numeric conversion
problem.
                        //                  customer_buf[i].c_balance = 0;
                        strcpy(customer_buf[i].c_balance,"");

                        customer_buf[i].c_ytd_payment = 0;
                        customer_buf[i].c_payment_cnt = 0;
                        customer_buf[i].c_delivery_cnt = 0;

                        strcpy(customer_buf[i].c_data,"");

                        customer_buf[i].h_amount = 0;

                        strcpy(customer_buf[i].h_data,"");

                }


}


//============================================================
============
//
// Function   : CustomerBufLoad
//
// Fills shared buffer for HISTORY and CUSTOMER
//============================================================
============

void CustomerBufLoad(int d_id, int w_id)
{
                long                                                    i;
   CUSTOMER_SORT_STRUCT    c[CUSTOMERS_PER_DISTRICT];


                for (i=0;i<customers_per_district;i++)
                {
                        if (i < 1000)
                                LastName(i, c[i].c_last);
                        else
                                LastName(NURand(255,0,999,LOADER_NURAND_C), c[i].c_last);

                                MakeAlphaString(8,16,FIRST_NAME_LEN,
c[i].c_first);

                                c[i].c_id = i+1;

                }

                printf("...Loading customer buffer for: d_id = %d, w_id = %d\n",
                                        d_id, w_id);

                for (i=0;i<customers_per_district;i++)
                {

                        customer_buf[i].c_d_id = d_id;
                        customer_buf[i].c_w_id = w_id;
                        customer_buf[i].h_amount = 10.0;

                        customer_buf[i].c_ytd_payment  = 10.0;

                        customer_buf[i].c_payment_cnt  = 1;
                        customer_buf[i].c_delivery_cnt = 0;

                        // Generate CUSTOMER and HISTORY data

                        customer_buf[i].c_id = c[i].c_id;

                        strcpy(customer_buf[i].c_first, c[i].c_first);
                        strcpy(customer_buf[i].c_last, c[i].c_last);

                        customer_buf[i].c_middle[0] = 'O';
                        customer_buf[i].c_middle[1] = 'E';

                        MakeAddress(customer_buf[i].c_street_1,

customer_buf[i].c_street_2,
                                                customer_buf[i].c_city,
                                                customer_buf[i].c_state,
                                                customer_buf[i].c_zip);

                        MakeNumberString(16, 16, PHONE_LEN,
customer_buf[i].c_phone);

                        if (RandomNumber(1L, 100L) > 10)
                                customer_buf[i].c_credit[0] = 'G';
                        else
                                customer_buf[i].c_credit[0] = 'B';
                        customer_buf[i].c_credit[1] = 'C';

                        customer_buf[i].c_credit_lim = 50000.0;
                        customer_buf[i].c_discount = ((float)
RandomNumber(0L, 5000L)) / 10000.0;

                        //  fix to avoid ODBC float to numeric conversion
problem.
                        //  customer_buf[i].c_balance = -10.0;
                        strcpy(customer_buf[i].c_balance,"-10.0");

                        MakeAlphaString(500, 500, C_DATA_LEN,
customer_buf[i].c_data);

                        // Generate HISTORY data
                        MakeAlphaString(12, 24, H_DATA_LEN,
customer_buf[i].h_data);

                }
}
```

```
//============================================================
============
//
// Function   : LoadCustomerTable
//
//============================================================
============

void LoadCustomerTable(LOADER_TIME_STRUCT *customer_time_start)
{
        int     i;
    long    c_id;
    short   c_d_id;
    short   c_w_id;
    char    c_first[FIRST_NAME_LEN+1];
    char    c_middle[MIDDLE_NAME_LEN+1];
    char    c_last[LAST_NAME_LEN+1];
    char    c_street_1[ADDRESS_LEN+1];
    char    c_street_2[ADDRESS_LEN+1];
    char    c_city[ADDRESS_LEN+1];
    char    c_state[STATE_LEN+1];
    char    c_zip[ZIP_LEN+1];
    char    c_phone[PHONE_LEN+1];
    char    c_credit[CREDIT_LEN+1];
    double  c_credit_lim;
    double  c_discount;

        //  fix to avoid ODBC float to numeric conversion problem.

        //  double              c_balance;
        char                    c_balance[6];

    double  c_ytd_payment;
    short   c_payment_cnt;
    short   c_delivery_cnt;
    char    c_data[C_DATA_LEN+1];
        char                    c_since[C_SINCE_LEN+1];
        RETCODE                 rc;

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

        rc = bcp_bind(c_hdbc1, (BYTE *) &c_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_first, 0, FIRST_NAME_LEN, NULL, 0, 0,
4);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_middle, 0, MIDDLE_NAME_LEN,NULL, 0,
0, 5);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_last, 0, LAST_NAME_LEN, NULL, 0, 0,
6);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_street_1, 0, ADDRESS_LEN, NULL, 0, 0,
7);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_street_2, 0, ADDRESS_LEN, NULL,0,0,
8);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_city, 0, ADDRESS_LEN, NULL, 0, 0, 9);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_state, 0, STATE_LEN, NULL, 0, 0, 10);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_zip, 0, ZIP_LEN, NULL, 0, 0, 11);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_phone, 0, PHONE_LEN, NULL, 0, 0, 12);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

        rc = bcp_bind(c_hdbc1, (BYTE *) &c_since, 0, C_SINCE_LEN,
NULL, 0, SQLCHARACTER, 13);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_credit, 0, CREDIT_LEN, NULL, 0, 0, 14);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_credit_lim, 0, SQL_VARLEN_DATA,
NULL, 0, SQLFLT8, 15);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_discount, 0, SQL_VARLEN_DATA,
NULL, 0, SQLFLT8, 16);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);


        //  fix to avoid ODBC float to numeric conversion problem.

    //  rc = bcp_bind(c_hdbc1, (BYTE *) &c_balance, 0, SQL_VARLEN_DATA,
NULL, 0, SQLFLT8, 17);
        //  if (rc != SUCCEED)
        //          HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_balance, 0, 5, NULL, 0,
SQLCHARACTER, 17);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);


    rc = bcp_bind(c_hdbc1, (BYTE *) &c_ytd_payment, 0, SQL_VARLEN_DATA,
NULL, 0, SQLFLT8, 18);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_payment_cnt, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 19);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) &c_delivery_cnt,0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 20);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *) c_data, 0, 500, NULL, 0, 0, 21);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

    for (i = 0; i < customers_per_district; i++)
    {
                c_id = customer_buf[i].c_id;
                c_d_id = customer_buf[i].c_d_id;
                c_w_id = customer_buf[i].c_w_id;

                strcpy(c_first, customer_buf[i].c_first);
                strcpy(c_middle, customer_buf[i].c_middle);
                strcpy(c_last, customer_buf[i].c_last);
                strcpy(c_street_1, customer_buf[i].c_street_1);
                strcpy(c_street_2, customer_buf[i].c_street_2);
                strcpy(c_city, customer_buf[i].c_city);
                strcpy(c_state, customer_buf[i].c_state);
                strcpy(c_zip, customer_buf[i].c_zip);
                strcpy(c_phone, customer_buf[i].c_phone);
                strcpy(c_credit, customer_buf[i].c_credit);

                FormatDate(&c_since);


                c_credit_lim = customer_buf[i].c_credit_lim;
                c_discount = customer_buf[i].c_discount;

                //  fix to avoid ODBC float to numeric conversion
problem.
                //  c_balance = customer_buf[i].c_balance;
                strcpy(c_balance, customer_buf[i].c_balance);

                c_ytd_payment = customer_buf[i].c_ytd_payment;
                c_payment_cnt = customer_buf[i].c_payment_cnt;
                c_delivery_cnt = customer_buf[i].c_delivery_cnt;

                strcpy(c_data, customer_buf[i].c_data);

                // Send data to server
                rc = bcp_sendrow(c_hdbc1);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc1);


                customer_rows_loaded++;
                CheckForCommit(c_hdbc1, c_hstmt1,
customer_rows_loaded, "customer", &customer_time_start->time_start);
    }

}


//============================================================
============
//
// Function   : LoadHistoryTable
//
//============================================================
============

void LoadHistoryTable(LOADER_TIME_STRUCT *history_time_start)
{
        int     i;
    long    c_id;
    short   c_d_id;
    short   c_w_id;
        double          h_amount;
    char    h_data[H_DATA_LEN+1];
        char                    h_date[H_DATE_LEN+1];
        RETCODE                 rc;
```

```
    rc = bcp_bind(c_hdbc2, (BYTE *) &c_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 4);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &c_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 5);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

                rc = bcp_bind(c_hdbc2, (BYTE *) &h_date, 0, H_DATE_LEN,
NULL, 0, SQLCHARACTER, 6);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) &h_amount, 0, SQL_VARLEN_DATA,
NULL, 0, SQLFLT8, 7);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *) h_data, 0, H_DATA_LEN, NULL, 0, 0, 8);
                if (rc != SUCCEED)
                        HandleErrorDBC(c_hdbc2);

    for (i = 0; i < customers_per_district; i++)
    {
                        c_id = customer_buf[i].c_id;
                        c_d_id = customer_buf[i].c_d_id;
                        c_w_id = customer_buf[i].c_w_id;
                        h_amount = customer_buf[i].h_amount;
                        strcpy(h_data, customer_buf[i].h_data);

                        FormatDate(&h_date);

                        // send to server
                        rc = bcp_sendrow(c_hdbc2);
                        if (rc != SUCCEED)
                                HandleErrorDBC(c_hdbc2);


                        history_rows_loaded++;
                        CheckForCommit(c_hdbc2, c_hstmt2,
history_rows_loaded, "history", &history_time_start->time_start);
    }

}


//============================================================
=-===============
//
// Function   : LoadOrders
//
//============================================================
=================

void LoadOrders()
{
                LOADER_TIME_STRUCT      orders_time_start;
                LOADER_TIME_STRUCT      new_order_time_start;
                LOADER_TIME_STRUCT      order_line_time_start;
                short                   w_id;
    short               d_id;
                DWORD
                dwThreadID[MAX_ORDER_THREADS];
                HANDLE
                hThread[MAX_ORDER_THREADS];
                char                    name[20];
                RETCODE
                rc;
                char
                bcphint[128];

                // seed with unique number
                seed(6);

                printf("Loading orders...\n");

                // if build index before load...
                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                                        BuildIndex("idxordcl");
                                        BuildIndex("idxnodcl");
                                        BuildIndex("idxodlcl");
                }

                // initialize bulk copy
                sprintf(name, "%s..%s", aptr->database, "orders");

                rc = bcp_init(o_hdbc1, name, NULL, "logs\\orders.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc1);

                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                        sprintf(bcphint, "tablock, order (o_w_id, o_d_id, o_id),
ROWS_PER_BATCH = %u", (aptr->num_warehouses * 30000));
                        rc = bcp_control(o_hdbc1, BCPHINTS, (void *)
bcphint);
                        if (rc != SUCCEED)
                                HandleErrorDBC(o_hdbc1);
                }

                sprintf(name, "%s..%s", aptr->database, "new_order");

                rc = bcp_init(o_hdbc2, name, NULL, "logs\\neword.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc2);

                if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                        sprintf(bcphint, "tablock, order (no_w_id, no_d_id,
no_o_id), ROWS_PER_BATCH = %u", (aptr->num_warehouses * 9000));
                        rc = bcp_control(o_hdbc2, BCPHINTS, (void *)
bcphint);
                        if (rc != SUCCEED)
                                HandleErrorDBC(o_hdbc2);
                }

                sprintf(name, "%s..%s", aptr->database, "order_line");

                rc = bcp_init(o_hdbc3, name, NULL, "logs\\ordline.err", DB_IN);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc3);

    if ((aptr->build_index == 1) && (aptr->index_order == 1))
                {
                        sprintf(bcphint, "tablock, order (ol_w_id, ol_d_id,
ol_o_id, ol_number), ROWS_PER_BATCH = %u", (aptr->num_warehouses *
300000));
                        rc = bcp_control(o_hdbc3, BCPHINTS, (void *)
bcphint);
                        if (rc != SUCCEED)
                                HandleErrorDBC(o_hdbc3);
                }

                orders_rows_loaded      = 0;
                new_order_rows_loaded   = 0;
                order_line_rows_loaded  = 0;

                OrdersBufInit();

                orders_time_start.time_start = (TimeNow() / MILLI);
                new_order_time_start.time_start = (TimeNow() / MILLI);
                order_line_time_start.time_start = (TimeNow() / MILLI);

                for (w_id = (short)aptr->starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
                {
                        for (d_id = 1; d_id <=
DISTRICT_PER_WAREHOUSE; d_id++)
                        {

                                OrdersBufLoad(d_id, w_id);

                                // start parallel loading threads here...

                                // start Orders table thread

                                printf("...Loading Order Table for: d_id =
%d, w_id = %d\n", d_id, w_id);

                                hThread[0] = CreateThread(NULL,

                                  0,

                                  (LPTHREAD_START_ROUTINE)
LoadOrdersTable,

                                  &orders_time_start,

                                  0,

                                  &dwThreadID[0]);

                                if (hThread[0] == NULL)
                                {
                                        printf("Error, failed in
creating creating thread = 0.\n");
                                        exit(-1);
                                }

                                // start NewOrder table thread

                                printf("...Loading New-Order Table for:
d_id = %d, w_id = %d\n", d_id, w_id);

                                hThread[1] = CreateThread(NULL,

                                  0,

                                  (LPTHREAD_START_ROUTINE)
LoadNewOrderTable,

                                  &new_order_time_start,

                                  0,
```

```
                                        &dwThreadID[1]);

                                if (hThread[1] == NULL)
                                {
                                        printf("Error, failed in
creating creating thread = 1.\n");
                                        exit(-1);
                                }

                                // start Order-Line table thread

                                printf("...Loading Order-Line Table for:
d_id = %d, w_id = %d\n", d_id, w_id);

                                hThread[2] = CreateThread(NULL,

                                        0,

                                        (LPTHREAD_START_ROUTINE)
LoadOrderLineTable,

                                        &order_line_time_start,

                                        0,

                                        &dwThreadID[2]);

                                if (hThread[2] == NULL)
                                {
                                        printf("Error, failed in
creating creating thread = 2.\n");
                                        exit(-1);
                                }

                                WaitForSingleObject( hThread[0],
INFINITE );
                                WaitForSingleObject( hThread[1],
INFINITE );
                                WaitForSingleObject( hThread[2],
INFINITE );

                                if (CloseHandle(hThread[0]) == FALSE)
                                {
                                        printf("Error, failed in
closing Orders thread handle with errno: %d\n", GetLastError());
                                }

                                if (CloseHandle(hThread[1]) == FALSE)
                                {
                                        printf("Error, failed in
closing NewOrder thread handle with errno: %d\n", GetLastError());
                                }

                                if (CloseHandle(hThread[2]) == FALSE)
                                {
                                        printf("Error, failed in
closing OrderLine thread handle with errno: %d\n", GetLastError());
                                }

                        }
                }

                printf("Finished loading orders.\n");


    return;
}
```

```
//=============================================================
============
//
// Function   : OrdersBufInit
//
// Clears shared buffer for ORDERS, NEWORDER, and ORDERLINE
//
//=============================================================
============

void OrdersBufInit()
{
        int     i;
        int                     j;

        for (i=0;i<orders_per_district;i++)
        {
                orders_buf[i].o_id = 0;
                orders_buf[i].o_d_id = 0;
                orders_buf[i].o_w_id = 0;
                orders_buf[i].o_c_id = 0;
                orders_buf[i].o_carrier_id = 0;
                orders_buf[i].o_ol_cnt = 0;
                orders_buf[i].o_all_local = 0;

                for (j=0;j<=14;j++)
                {
                        orders_buf[i].o_ol[j].ol = 0;
                        orders_buf[i].o_ol[j].ol_i_id = 0;
                        orders_buf[i].o_ol[j].ol_supply_w_id = 0;
                        orders_buf[i].o_ol[j].ol_quantity = 0;
                        orders_buf[i].o_ol[j].ol_amount = 0;

        strcpy(orders_buf[i].o_ol[j].ol_dist_info,"");
                }
        }



}


//=============================================================
============
//
// Function   : OrdersBufLoad
//
// Fills shared buffer for ORDERS, NEWORDER, and ORDERLINE
//
//=============================================================
============

void OrdersBufLoad(int d_id, int w_id)
{

        int     cust[ORDERS_PER_DIST+1];
        long    o_id;
        short   ol;

        printf("...Loading Order Buffer for: d_id = %d, w_id = %d\n",
                                d_id, w_id);

        GetPermutation(cust, ORDERS_PER_DIST);

        for (o_id=0;o_id<orders_per_district;o_id++)
        {
                // Generate ORDER and NEW-ORDER data

                orders_buf[o_id].o_d_id = d_id;
                orders_buf[o_id].o_w_id = w_id;
                orders_buf[o_id].o_id = o_id+1;
                orders_buf[o_id].o_c_id = cust[o_id+1];
                orders_buf[o_id].o_ol_cnt =
(short)RandomNumber(5L, 15L);

                if (o_id < first_new_order)
                {
                        orders_buf[o_id].o_carrier_id =
(short)RandomNumber(1L, 10L);
                        orders_buf[o_id].o_all_local  = 1;
                }
                else
                {
                        orders_buf[o_id].o_carrier_id = 0;
                        orders_buf[o_id].o_all_local  = 1;
                }

                for (ol=0; ol<orders_buf[o_id].o_ol_cnt; ol++)
                {

                        orders_buf[o_id].o_ol[ol].ol = ol+1;
                        orders_buf[o_id].o_ol[ol].ol_i_id =
RandomNumber(1L, max_items);

        orders_buf[o_id].o_ol[ol].ol_supply_w_id = w_id;
                        orders_buf[o_id].o_ol[ol].ol_quantity = 5;
                        MakeAlphaString(24, 24,
OL_DIST_INFO_LEN, &orders_buf[o_id].o_ol[ol].ol_dist_info);

                        // Generate ORDER-LINE data
                        if (o_id < first_new_order)
                        {

        orders_buf[o_id].o_ol[ol].ol_amount = 0;
                                                // Added to insure
ol_delivery_d set properly during load


                FormatDate(&orders_buf[o_id].o_ol[ol].ol_delivery_d);


                        }
                        else
                        {

        orders_buf[o_id].o_ol[ol].ol_amount =
RandomNumber(1,999999)/100.0;
                                                // Added to insure
ol_delivery_d set properly during load

                                                // odbc datetime format

        strcpy(orders_buf[o_id].o_ol[ol].ol_delivery_d,"1899-12-31
12:00:00.000");

                        }
                }
        }
}


//=============================================================
============
//
// Function   : LoadOrdersTable
//
//=============================================================
============

void LoadOrdersTable(LOADER_TIME_STRUCT *orders_time_start)
{
```

```
        int     i;
long    o_id;
        short   o_d_id;
        short   o_w_id;
long    o_c_id;
short   o_carrier_id;
short   o_ol_cnt;
short   o_all_local;
        char                    o_entry_d[O_ENTRY_D_LEN+1];
        RETCODE                 rc;
        DBINT                   rcint;

        // bind ORDER data
rc = bcp_bind(o_hdbc1, (BYTE *) &o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

rc = bcp_bind(o_hdbc1, (BYTE *) &o_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

rc = bcp_bind(o_hdbc1, (BYTE *) &o_w_id, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

rc = bcp_bind(o_hdbc1, (BYTE *) &o_c_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 4);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        rc = bcp_bind(o_hdbc1, (BYTE *) &o_entry_d, 0,
O_ENTRY_D_LEN, NULL, 0, SQLCHARACTER, 5);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

rc = bcp_bind(o_hdbc1, (BYTE *) &o_carrier_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 6);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

rc = bcp_bind(o_hdbc1, (BYTE *) &o_ol_cnt, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 7);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

rc = bcp_bind(o_hdbc1, (BYTE *) &o_all_local, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 8);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

for (i = 0; i < orders_per_district; i++)
{
                o_id        = orders_buf[i].o_id;
                o_d_id      = orders_buf[i].o_d_id;
                o_w_id      = orders_buf[i].o_w_id;
                o_c_id      = orders_buf[i].o_c_id;
                o_carrier_id = orders_buf[i].o_carrier_id;
                o_ol_cnt    = orders_buf[i].o_ol_cnt;
                o_all_local  = orders_buf[i].o_all_local;

                FormatDate(&o_entry_d);

                // send data to server
                rc = bcp_sendrow(o_hdbc1);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc1);

                orders_rows_loaded++;
                CheckForCommit(o_hdbc1, o_hstmt1,
orders_rows_loaded, "orders", &orders_time_start->time_start);
}

        // rcint = bcp_batch(o_hdbc1);
        // if (rcint < 0)
        //      HandleErrorDBC(o_hdbc1);

        if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
        {
                rcint = bcp_done(o_hdbc1);
                if (rcint < 0)
                        HandleErrorDBC(o_hdbc1);


                SQLFreeStmt(o_hstmt1, SQL_DROP);
                SQLDisconnect(o_hdbc1);
                SQLFreeConnect(o_hdbc1);

                // if build index after load...
                if ((aptr->build_index == 1) && (aptr->index_order ==
0))
                        BuildIndex("idxordcl");

                // build non-clustered index
                if (aptr->build_index == 1)
                        BuildIndex("idxordnc");
        }

}


//=============================================================
============
//
// Function   : LoadNewOrderTable
//
//=============================================================
============

void LoadNewOrderTable(LOADER_TIME_STRUCT *new_order_time_start)
{
int     i;
long    o_id;
short   o_d_id;
short   o_w_id;
        RETCODE                 rc;
        DBINT                   rcint;

        // Bind NEW-ORDER data

rc = bcp_bind(o_hdbc2, (BYTE *) &o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

rc = bcp_bind(o_hdbc2, (BYTE *) &o_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

rc = bcp_bind(o_hdbc2, (BYTE *) &o_w_id, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

        for (i = first_new_order; i < last_new_order; i++)
        {
                o_id    = orders_buf[i].o_id;
                o_d_id  = orders_buf[i].o_d_id;
                o_w_id  = orders_buf[i].o_w_id;

                rc = bcp_sendrow(o_hdbc2);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc2);


                new_order_rows_loaded++;

                CheckForCommit(o_hdbc2, o_hstmt2,
new_order_rows_loaded, "new_order", &new_order_time_start->time_start);
        }

        // rcint = bcp_batch(o_hdbc2);
        // if (rcint < 0)
        //      HandleErrorDBC(o_hdbc2);

        if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
        {
                rcint = bcp_done(o_hdbc2);
                if (rcint < 0)
                        HandleErrorDBC(o_hdbc2);


                SQLFreeStmt(o_hstmt2, SQL_DROP);
                SQLDisconnect(o_hdbc2);
                SQLFreeConnect(o_hdbc2);

                // if build index after load...
                if ((aptr->build_index == 1) && (aptr->index_order ==
0))
                        BuildIndex("idxnodcl");

        }

}


//=============================================================
============
//
// Function   : LoadOrderLineTable
//
//=============================================================
============

void LoadOrderLineTable(LOADER_TIME_STRUCT *order_line_time_start)
{
        int     i,j;
long    o_id;
        short   o_d_id;
        short   o_w_id;
long    ol;
        long    ol_i_id;
short   ol_supply_w_id;
short   ol_quantity;
double  ol_amount;
char    ol_dist_info[DIST_INFO_LEN+1];
        char
        ol_delivery_d[OL_DELIVERY_D_LEN+1];
        RETCODE                 rc;
        DBINT                   rcint;

        // bind ORDER-LINE data
rc = bcp_bind(o_hdbc3, (BYTE *) &o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc3);

rc = bcp_bind(o_hdbc3, (BYTE *) &o_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
```

```
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) &o_w_id, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 3);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) &ol, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 4);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) &ol_i_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 5);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) &ol_supply_w_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 6);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

		rc = bcp_bind(o_hdbc3, (BYTE *) &ol_delivery_d, 0,
OL_DELIVERY_D_LEN, NULL, 0, SQLCHARACTER, 7);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) &ol_quantity, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 8);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) &ol_amount, 0, SQL_VARLEN_DATA,
NULL, 0, SQLFLT8, 9);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *) ol_dist_info, 0, DIST_INFO_LEN, NULL, 0,
0, 10);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc3);

		for (i = 0; i < orders_per_district; i++)
	{
			o_id    = orders_buf[i].o_id;
			o_d_id  = orders_buf[i].o_d_id;
			o_w_id  = orders_buf[i].o_w_id;

			for (j=0; j < orders_buf[i].o_ol_cnt; j++)
			{
				ol          = orders_buf[i].o_ol[j].ol;
				ol_i_id     =
orders_buf[i].o_ol[j].ol_i_id;
				ol_supply_w_id =
orders_buf[i].o_ol[j].ol_supply_w_id;
				ol_quantity    =
orders_buf[i].o_ol[j].ol_quantity;
				ol_amount      =
orders_buf[i].o_ol[j].ol_amount;

		strcpy(ol_delivery_d,orders_buf[i].o_ol[j].ol_delivery_d);


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

				rc = bcp_sendrow(o_hdbc3);
				if (rc != SUCCEED)


		HandleErrorDBC(o_hdbc3);
```

```
				order_line_rows_loaded++;
				CheckForCommit(o_hdbc3, o_hstmt3,
order_line_rows_loaded, "order_line", &order_line_time_start->time_start);
			}

		}

		// rcint = bcp_batch(o_hdbc3);
		// if (rcint < 0)
		//	HandleErrorDBC(o_hdbc3);

		if ((o_w_id == aptr->num_warehouses) && (o_d_id == 10))
		{
			rcint = bcp_done(o_hdbc3);
			if (rcint < 0)
				HandleErrorDBC(o_hdbc3);


			SQLFreeStmt(o_hstmt3, SQL_DROP);
			SQLDisconnect(o_hdbc3);
			SQLFreeConnect(o_hdbc3);


			// if build index after load...
			if ((aptr->build_index == 1) && (aptr->index_order ==
0))
				BuildIndex("idxodlcl");

		}

}


//============================================================
============
//
// Function   : GetPermutation
//
//============================================================
============

void GetPermutation(int perm[], int n)
{
		int i, r, t;

		for (i=1;i<=n;i++)
			perm[i] = i;

		for (i=1;i<=n;i++)
		{
			r = RandomNumber(i,n);
			t = perm[i];
			perm[i] = perm[r];
			perm[r] = t;
		}

}


//============================================================
============
//
// Function   : CheckForCommit
//
//============================================================
============

void CheckForCommit(HDBC hdbc,
HSTMT
hstmt,
int rows_loaded,
```

```
char
*table_name,
long *time_start)
{

	long		time_end, time_diff;
			// DBINT	rcint;

	if ( !(rows_loaded % aptr->batch) )
	{

			// rcint = bcp_batch(hdbc);
			// if (rcint < 0)
			//	HandleErrorDBC(hdbc);

			time_end = (TimeNow() / MILLI);
			time_diff = time_end - *time_start;

			printf("-> Loaded %ld rows into %s in %ld sec - Total
= %d (%.2f rps)\n",
				aptr->batch,
				table_name,
					time_diff,
				rows_loaded,
					(float) aptr->batch /
(time_diff ? time_diff : 1L));

			*time_start = time_end;
	}

	return;
}


//============================================================
============
//
// Function   : OpenConnections
//
//============================================================
============

void OpenConnections()
{

		RETCODE		rc;

		char			szDriverString[300];
		char			szDriverStringOut[1024];
		SQLSMALLINT		cbDriverStringOut;


		SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE,
&henv );

		SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0 );

		SQLAllocHandle(SQL_HANDLE_DBC, henv , &i_hdbc1);
		SQLAllocHandle(SQL_HANDLE_DBC, henv , &w_hdbc1);
		SQLAllocHandle(SQL_HANDLE_DBC, henv , &c_hdbc1);
		SQLAllocHandle(SQL_HANDLE_DBC, henv , &c_hdbc2);
		SQLAllocHandle(SQL_HANDLE_DBC, henv , &o_hdbc1);
		SQLAllocHandle(SQL_HANDLE_DBC, henv , &o_hdbc2);
		SQLAllocHandle(SQL_HANDLE_DBC, henv , &o_hdbc3);

		SQLSetConnectAttr(i_hdbc1, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );
		SQLSetConnectAttr(w_hdbc1, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );
```

```
		SQLSetConnectAttr(c_hdbc1, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );
		SQLSetConnectAttr(c_hdbc2, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );
		SQLSetConnectAttr(o_hdbc1, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );
		SQLSetConnectAttr(o_hdbc2, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );
		SQLSetConnectAttr(o_hdbc3, SQL_COPT_SS_BCP, (void
*)SQL_BCP_ON, SQL_IS_INTEGER );

		// Open connections to SQL Server

		// Connection 1

		sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

		rc = SQLSetConnectOption (i_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
		if (rc != SUCCEED)
			HandleErrorDBC(i_hdbc1);

		rc = SQLDriverConnect ( i_hdbc1,
                                                                NULL,

		(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

		(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
		if (rc != SUCCEED)
			HandleErrorDBC(i_hdbc1);

		// Connection 2

		sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

		rc = SQLSetConnectOption (w_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		rc = SQLDriverConnect ( w_hdbc1,

		NULL,

		(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

		(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
		if (rc != SUCCEED)
			HandleErrorDBC(w_hdbc1);

		// Connection 3

		sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

		rc = SQLSetConnectOption (c_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
		if (rc != SUCCEED)
			HandleErrorDBC(c_hdbc1);

		rc = SQLDriverConnect ( c_hdbc1,

		NULL,

		(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

		(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
		if (rc != SUCCEED)
			HandleErrorDBC(c_hdbc1);

		// Connection 4

		sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

		rc = SQLSetConnectOption (c_hdbc2, SQL_PACKET_SIZE, aptr-
>pack_size);
		if (rc != SUCCEED)
			HandleErrorDBC(c_hdbc2);

		rc = SQLDriverConnect ( c_hdbc2,

		NULL,

		(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

		(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
		if (rc != SUCCEED)
			HandleErrorDBC(c_hdbc2);

		// Connection 5

		sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

		rc = SQLSetConnectOption (o_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc1);

		rc = SQLDriverConnect ( o_hdbc1,

		NULL,

		(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

		(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc1);

		// Connection 6

		sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

		rc = SQLSetConnectOption (o_hdbc2, SQL_PACKET_SIZE, aptr-
>pack_size);
		if (rc != SUCCEED)
			HandleErrorDBC(o_hdbc2);

		rc = SQLDriverConnect ( o_hdbc2,
```

```
                NULL,

                (SQLCHAR*)&szDriverString[0] ,

                SQL_NTS,

                (SQLCHAR*)&szDriverStringOut[0],

                sizeof(szDriverStringOut),

                &cbDriverStringOut,

                SQL_DRIVER_NOPROMPT );
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc2);

                // Connection 7

                sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

aptr->server,

aptr->user,

aptr->password,

aptr->database );

                rc = SQLSetConnectOption (o_hdbc3, SQL_PACKET_SIZE, aptr-
>pack_size);
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc3);

                rc = SQLDriverConnect ( o_hdbc3,

                NULL,

                (SQLCHAR*)&szDriverString[0] ,

                SQL_NTS,

                (SQLCHAR*)&szDriverStringOut[0],

                sizeof(szDriverStringOut),

                &cbDriverStringOut,

                SQL_DRIVER_NOPROMPT );
                if (rc != SUCCEED)
                        HandleErrorDBC(o_hdbc3);

}


//============================================================
============
//
// Function name: BuildIndex
//
//============================================================
============

void BuildIndex(char        *index_script)
{
                char            cmd[256];

                printf("Starting index creation:  %s\n",index_script);

                sprintf(cmd, "isql -S%s -U%s -P%s -e -i%s\\%s.sql > logs\\%s.log",
                                                        aptr->server,
                                                        aptr->user,
                                                        aptr->password,
                                                        aptr->index_script_path,
                                                        index_script,
                                                        index_script);

                system(cmd);

                printf("Finished index creation:  %s\n",index_script);
}


void HandleErrorDBC (SQLHDBC  hdbc1)
{

                SQLCHAR                 SqlState[6],
Msg[SQL_MAX_MESSAGE_LENGTH];
                SQLINTEGER  NativeError;
                SQLSMALLINT i, MsgLen;
                SQLRETURN   rc2;
                char                    timebuf[128];
                char                    datebuf[128];
                FILE                    *fp1;

                i = 1;
                while (( rc2 = SQLGetDiagRec(SQL_HANDLE_DBC , hdbc1, i,
SqlState , &NativeError,
                                                        Msg, sizeof(Msg) ,
&MsgLen )) != SQL_NO_DATA )
                {

                        sprintf( szLastError , "%s" ,  Msg );

                        _strtime(timebuf);
                        _strdate(datebuf);

                        printf( "[%s : %s] %s\n" , datebuf, timebuf,
szLastError);

                        fp1 = fopen("logs\\tpccldr.err","w");
                        if (fp1 == NULL)
                                printf("ERROR:  Unable to open errorlog
file.\n");
                        else
                        {
                                fprintf(fp1, "[%s : %s] %s\n" , datebuf,
timebuf, szLastError);
                                fclose(fp1);
                        }

                        i++;

                }
}


void FormatDate ( char* szTimeCOutput )
{

                struct tm when;
                time_t now;

                time( &now );
                when  = *localtime( &now );

                mktime( &when );

                // odbc datetime format
                strftime( szTimeCOutput , 30 , "%Y-%m-%d %H:%M:%S.000",
&when );

                return;
}
```

Appendix C : Tunable Parameters

RTE input parameter

The following parameters were used with Microsoft BenchCraft RTE..

Profile: 1478
File Path: C:\benchcrf\1478.pro
Version: 1.0.1

Number of Engines: 15

Name: DRIVER1
Description: rte11
Directory: \drv11.OUT
Machine: RTE1
Parameter Set: TPCC
Index: 0
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER112682656
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER10
Description: rte52
Directory: \drv52.out
Machine: rte5
Parameter Set: TPCC
Index: 900000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER10579000
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER11
Description: rte13
Directory: \drv13.out
Machine: rte1
Parameter Set: TPCC
Index: 1000000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER11615140
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER12
Description: rte23
Directory: \drv23.out
Machine: rte2
Parameter Set: TPCC
Index: 1100000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER12647093
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER13
Description: rte33
Directory: \drv33.out
Machine: rte3
Parameter Set: TPCC
Index: 1200000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER13673953
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER14
Description: rte43
Directory: \drv43.out
Machine: rte4
Parameter Set: TPCC
Index: 1300000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER14699187
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER15
Description: rte53
Directory: \drv53.out
Machine: rte5
Parameter Set: TPCC
Index: 1400000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER15721281
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER2
Description: rte21
Directory: \drv21.OUT
Machine: RTE2
Parameter Set: TPCC
Index: 100000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER212716915
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER3
Description: rte31
Directory: \drv31.OUT
Machine: RTE3
Parameter Set: TPCC
Index: 200000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER312737245
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER4
Description: rte41
Directory: \drv41.OUT
Machine: RTE4
Parameter Set: TPCC
Index: 300000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER412751255
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER5
Description: rte51
Directory: \drv51.OUT
Machine: RTE5
Parameter Set: TPCC
Index: 400000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER512764564
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER6
Description: rte12
Directory: \drv12.out
Machine: rte1
Parameter Set: TPCC
Index: 500000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER6391906
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER7
Description: rte22
Directory: \drv22.out
Machine: rte2
Parameter Set: TPCC
Index: 600000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER7431859
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER8
Description: rte32
Directory: \drv32.out
Machine: rte3
Parameter Set: TPCC
Index: 700000000
Seed: 1473
Configured Users: 990
Pipe Name: DRIVER8468046
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Name: DRIVER9
Description: rte42
Directory: \drv42.out
Machine: rte4
Parameter Set: TPCC
Index: 800000000
Seed: 1473
Configured Users: 980
Pipe Name: DRIVER9519828
Connect Rate: 300
Start Rate: 0
CLIENT_NURAND: 233
CPU: 0

Number of User groups: 15

Driver Engine: DRIVER1
IIS Server: client011
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 1 - 99
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER13
IIS Server: client033
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 791 - 888
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER4
IIS Server: client041
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 889 - 987
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER9
IIS Server: client042
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 988 - 1085
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER14
IIS Server: client043
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 1086 - 1183
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER6
IIS Server: client012
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 100 - 198
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER11
IIS Server: client013
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 199 - 296
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER2
IIS Server: client021
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 297 - 395
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER7
IIS Server: client022
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 396 - 494
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER5
IIS Server: client051
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 1184 - 1282
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER10
IIS Server: client052
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 1283 - 1380
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER12
IIS Server: client023
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 495 - 592
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER15
IIS Server: client053
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 1381 - 1478
w_id Max Warehouse: 1478
Scale: Normal
User Count: 980
District id: 1
Scale Down: No

Driver Engine: DRIVER3
IIS Server: CLIENT031
SQL Server: TPCC_99F
User: sa
Protocol: Html
w_id Range: 593 - 691
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Driver Engine: DRIVER8
IIS Server: client032
SQL Server: tpcc_99f
User: sa
Protocol: Html
w_id Range: 692 - 790
w_id Max Warehouse: 1478
Scale: Normal
User Count: 990
District id: 1
Scale Down: No

Number of Parameter Sets: 2

TPCC

		Txn Weight	Think Time	Key Time	RT Delay	RT Fence	Menu
New Order		44.80	12.07			18.01	
	0.10	5.00	0.10				
Payment		43.08	12.07			3.01	
	0.10	5.00	0.10				
Delivery		4.04	5.07			2.01	
	0.10	5.00	0.10				
Stock Level		4.04	5.07			2.01	
	0.10	20.00	0.10				

```
                    Order Status      4.04        10.07         2.01
0.10        5.00      0.10

                    ~Default
                    Default Parameter Set
                              Txn    Think   Key    RT     RT     Menu
                             Weight  Time    Time   Delay  Fence
Delay
                    New Order       10.00       12.05         18.01
0.10        5.00      0.10
                    Payment         10.00       12.05         3.01
0.10        5.00      0.10
                    Delivery        1.00        5.05          2.01
0.10        5.00      0.10
                    Stock Level     1.00        5.05          2.01
0.10        20.00     0.10
                    Order Status    1.00        10.05         2.01
0.10        5.00      0.10
```

<Server Configuration>

Microsoft Windows NT Server version 4.0 Configuration Parameters

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

-Alerter
-Computer Browser
-License Logging Service
-Messenger
-NT LM Security Support Provider
-Plug and Play
-Server
-Spooler
-TCP/IP Netbios Helper
-Workstation

BOOT. INI

The /3gb switch was added to the boot. ini file to cause NT Enterprise Server to allow 3GB of user and 1GB of kernel virtual address space, rather than the usual 2GB of virtual address space.

NT Registry

No Windows NT registry parameters were modified for this benckmark.

System Configuration Report

```
Microsoft Diagnostics Report For \\TPCC_99F
------------------------------------------------------------------------

OS Version Report
------------------------------------------------------------------------
Microsoft (R) Windows NT (TM) Server
Version 4.0 (Build 1381: Service Pack 3) x86 Multiprocessor Free
Registered Owner: 3dev, nec
Product Number: 70238-415-0002856-69424
------------------------------------------------------------------------

System Report
------------------------------------------------------------------------
System: AT/AT COMPATIBLE
Hardware Abstraction Layer: MPS 1.4 - APIC platform
BIOS Date: 07/15/98
BIOS Version: PhoenixBIOS 4.0 Release 6.0.0316

Processor list:
  0:  x86 Family 6 Model 5 Stepping 2 GenuineIntel ~399 Mhz
  1:  x86 Family 6 Model 5 Stepping 2 GenuineIntel ~399 Mhz
  2:  x86 Family 6 Model 5 Stepping 2 GenuineIntel ~399 Mhz
  3:  x86 Family 6 Model 5 Stepping 2 GenuineIntel ~399 Mhz
------------------------------------------------------------------------

Video Display Report
------------------------------------------------------------------------
BIOS Date: 06/25/98
BIOS Version: CL-GD546x Laguna PCI VGA BIOS Version 1.71e c15

Adapter:
  Setting: 1024 x 768 x 256
           60 Hz
  Type: cl546xm compatible display adapter
  String: Cirrus Logic VisualMedia(TM) Accelerator
  Memory: 2 MB
  Chip Type: Cirrus Logic 5465
  DAC Type: Internal
Driver:
  Vendor: Cirrus Logic, Inc.
  File(s): cl546xm.sys, cl5465.dll
  Version: 4.00.1381.170g-nt170g.eng01, 4.0.1


Drives Report
------------------------------------------------------------------------
C:\  (Local - NTFS)  Total: 4,690,976KB, Free: 4,576,452KB
  Serial Number: 84AB - 56D0
  Bytes per cluster: 512
  Sectors per cluster: 8
  Filename length: 255
D:\  (Local - NTFS)  Total: 0KB, Free: 0KB
  Serial Number: A83B - 365C
  Bytes per cluster: 512
  Sectors per cluster: 1
  Filename length: 255
W:\  (Local - NTFS) tpccback Total: 124,407,744KB, Free: 6,674,496KB
  Serial Number: C0D8 - 37BF
  Bytes per cluster: 512
  Sectors per cluster: 128
  Filename length: 255

Memory Report
------------------------------------------------------------------------
Handles: 901
Threads: 76
Processes: 14

Physical Memory (K)
  Total: 4,128,172
  Available: 3,942,452
  File Cache: 16,308

Kernel Memory (K)
  Total: 13,440
  Paged: 7,776
  Nonpaged: 5,664

Commit Charge (K)
  Total: 25,296
  Limit: 6,074,972
  Peak: 2,862,384

Pagefile Space (K)
  Total: 2,108,416
  Total in use: 2,160
  Peak: 2,472

  D:\pagefile.sys
    Total: 2,108,416
    Total in use: 2,160
    Peak: 2,472


Services Report
------------------------------------------------------------------------
Alerter                         Stopped   (Manual)
  D:\WINNT\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
    LanmanWorkstation
Computer Browser                Stopped   (Manual)
  D:\WINNT\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
    LanmanWorkstation
    LanmanServer
    LmHosts
ClipBook Server                 Stopped   (Manual)
  D:\WINNT\system32\clipsrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
    NetDDE
DHCP Client (TDI)               Stopped   (Disabled)
  D:\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)
  D:\WINNT\system32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
GAM Server Services             Stopped   (Manual)
  D:\WINNT\SYSTEM32\GAMSERV\gamscm.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
Server                          Stopped   (Manual)
  D:\WINNT\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Group Dependencies:
```

```
    TDI
Workstation (NetworkProvider)              Stopped   (Manual)
  D:\WINNT\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Group Dependencies:
    TDI
License Logging Service                    Stopped   (Manual)
  D:\WINNT\System32\llssrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
TCP/IP NetBIOS Helper                      Stopped   (Manual)
  D:\WINNT\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Group Dependencies:
    NetworkProvider
Messenger                                  Stopped   (Manual)
  D:\WINNT\System32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
    LanmanWorkstation
    NetBios
MSDTC (MS Transactions)                    Stopped   (Manual)
  D:\WINNT\System32\msdtc.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
  Service Dependencies:
    RPCSS
MSSQLServer                                Stopped   (Manual)
  D:\MSSQL7\binn\sqlservr.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
Network DDE (NetDDEGroup)                  Stopped   (Manual)
  D:\WINNT\system32\netdde.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
    NetDDEDSDM
Network DDE DSDM                           Stopped   (Manual)
  D:\WINNT\system32\netdde.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Net Logon (RemoteValidation)               Stopped   (Manual)
  D:\WINNT\System32\lsass.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
  Service Dependencies:
    LanmanWorkstation
    LmHosts
NT LM Security Support Provider            Stopped   (Manual)
  D:\WINNT\System32\SERVICES.EXE
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Plug and Play (PlugPlay)                   Stopped   (Manual)
  D:\WINNT\system32\services.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Shared Process
Protected Storage                          Stopped   (Manual)
  D:\WINNT\System32\pstores.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
  Service Dependencies:
    RpcSs
Directory Replicator                       Stopped   (Manual)
  D:\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)
  D:\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)
  D:\WINNT\system32\RpcSs.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
Schedule                                   Stopped   (Manual)
  D:\WINNT\System32\AtSvc.Exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
SNMP                                       Stopped   (Manual)
  D:\WINNT\System32\snmp.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
    Tcpip
    EventLog
SNMP Trap Service                          Stopped   (Manual)
  D:\WINNT\System32\snmptrap.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
    Tcpip
    EventLog
Spooler (SpoolerGroup)                     Stopped   (Manual)
  D:\WINNT\system32\spoolss.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process, Interactive
SQLServerAgent                             Stopped   (Manual)
  D:\MSSQL7\binn\sqlagent.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
  Service Dependencies:
    MSSQLServer
Telephony Service                          Stopped   (Manual)
  D:\WINNT\system32\tapisrv.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
UPS                                        Stopped   (Manual)
  D:\WINNT\System32\ups.exe
  Service Account Name: LocalSystem
  Error Severity: Normal
  Service Flags: Own Process
```

Drivers Report

```
Abiosdsk (Primary disk)                    Stopped   (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver, Shared Process
AFD Networking Support Environment (TDI)   Running   (Automatic)
  D:\WINNT\System32\drivers\afd.sys
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
Aha154x (SCSI miniport)                    Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
Aha174x (SCSI miniport)                    Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
aic78xx (SCSI miniport)                    Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
Always (SCSI miniport)                     Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
ami0nt (SCSI miniport)                     Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
amsint (SCSI miniport)                     Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
Arrow (SCSI miniport)                      Stopped   (Disabled)
  Error Severity: Normal
  Service Flags: Kernel Driver, Shared Process
atapi (SCSI miniport)                      Stopped   (Disabled)
  D:\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)                             Stopped   (Disabled)
  Error Severity: Ignore
  Service Flags: Kernel Driver, Shared Process
cl546x (Video)                             Running   (System)
```

```
   System32\DRIVERS\cl546xm.sys
   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)                Running   (Boot)
   D:\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
3Com 3C90x Adapter Driver (NDIS)        Running   (Automatic)
   D:\WINNT\System32\drivers\el90xnd4.sys
   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)                         Running   (Boot)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
gamdrv (SCSI Class)                     Stopped   (Disabled)
   D:\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)                        Running   (Boot)
   D:\WINNT\System32\drivers\macdisk.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
mainte (Extended Base)                  Stopped   (Disabled)
   D:\WINNT\System32\drivers\mainte.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
Msfs (File system)                      Running   (System)
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
Mup (Network)                           Stopped   (Manual)
   D:\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)        Stopped   (Manual)
   D:\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)
   D:\WINNT\System32\drivers\netbt.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
   Service Dependencies:
      Tcpip
NetDetect                               Stopped   (Manual)
   D:\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)                           Stopped   (Manual)
   D:\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)                    Running   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Serial (Extended base)                   Running   (Automatic)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sermouse (Pointer Port)                  Stopped   (Disabled)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
Sfloppy (Primary disk)                   Stopped   (System)
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
   Group Dependencies:
      SCSI miniport
Simbad (Filter)                          Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
slcd32 (SCSI miniport)                   Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Sparrow (SCSI miniport)                  Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Spock (SCSI miniport)                    Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
Srv (Network)                            Stopped   (Manual)
   D:\WINNT\System32\drivers\srv.sys
   Error Severity: Normal
   Service Flags: File System Driver, Shared Process
symc810 (SCSI miniport)                  Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
symc8xx (SCSI miniport)                  Running   (Boot)
   D:\WINNT\system32\drivers\symc8xx.sys
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T128 (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
T13B (SCSI miniport)                     Stopped   (Disabled)
   Error Severity: Normal
   Service Flags: Kernel Driver, Shared Process
TCP/IP Service (PNP_TDI)                 Running   (Automatic)
   D:\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)                     Running   (System)
   D:\WINNT\System32\drivers\vga.sys
   Error Severity: Ignore
   Service Flags: Kernel Driver, Shared Process
VgaStart (Video Init)                    Stopped   (System)
   D:\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
El90x                        44    44 0x00000000
Floppy                        6     6 0x00000000
dac960nt                      4     4 0x00000000
dac960nt                      8     8 0x00000000
dac960nt                     12    12 0x00000000
dac960nt                      4     4 0x00000000
dac960nt                      8     8 0x00000000
dac960nt                     12    12 0x00000000
dac960nt                     16    16 0x00000000
dac960nt                      4     4 0x00000000
dac960nt                      8     8 0x00000000
dac960nt                     12    12 0x00000000
symc8xx                      48    48 0x00000000
symc8xx                      48    48 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
El90x                        0x00004400  0x0000000080
Floppy                       0x000003f0  0x0000000006
Floppy                       0x000003f7  0x0000000001
symc8xx                       0x00004000  0x0000000100
symc8xx                       0x00005000  0x0000000100
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       0xfec01000  0x00000400
MPS 1.4 - APIC platform       0xfee00000  0x00000400
EI90x                    0xfe011000  0x00000080
dac960nt                   0xfe220000  0x00002000
dac960nt                   0xfe222000  0x00002000
dac960nt                   0xfe224000  0x00002000
dac960nt                   0xfe600000  0x00002000
dac960nt                   0xfe602000  0x00002000
dac960nt                   0xfe604000  0x00002000
dac960nt                   0xfe606000  0x00002000
dac960nt                   0xfea00000  0x00002000
dac960nt                   0xfea02000  0x00002000
dac960nt                   0xfea04000  0x00002000
symc8xx                    0xfe011400  0x00000100
symc8xx                    0xfe010000  0x00001000
symc8xx                    0xfe301000  0x00000100
symc8xx                    0xfe300000  0x00001000
cl546x                   0xfe000000  0x0000a000
cl546x                   0xfc000000  0x02000000
VgaSave                     0x000a0000  0x00020000


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


System Environment Variables
  ComSpec=D:\WINNT\system32\cmd.exe
  Os2LibPath=D:\WINNT\system32\os2\dll;
  Path=D:\WINNT\system32;D:\WINNT;;D:\MSSQL7\BINN
  windir=D:\WINNT
  OS=Windows_NT
  PROCESSOR_ARCHITECTURE=x86
  PROCESSOR_LEVEL=6
  PROCESSOR_IDENTIFIER=x86 Family 6 Model 5 Stepping 2, GenuineIntel
  PROCESSOR_REVISION=0502
  NUMBER_OF_PROCESSORS=4


Environment Variables for Current User
  TEMP=D:\TEMP
  TMP=D:\TEMP
```

<Client Configuration>

Tuxedo Configuration

This configuration file is used on each of the other 4 clients with the exception of the Hostname CLIENT01, which is replaced by CLIENT02 through CLIENT05.

```
#ident          "@(#)apps:simpapp/ubbsimple             60.3"
#Skeleton UBBCONFIG file for the TUXEDO Simple Application.
#Replace the <bracketed> items with the appropriate values.

*RESOURCES
IPCKEY                  123456
DOMAINID    tpcc
MASTER                  tpcc
MAXACCESSERS            512
MAXSERVERS              128
MAXSERVICES             512
MODEL                   SHM
LDBAL                   N

*MACHINES
DEFAULT:
APPDIR="c:\tux_ap"
TUXCONFIG="c:\tux_ap\tuxconfig"
TUXDIR="c:\tuxedo"
CLIENT01                LMID=tpcc

*GROUPS
GROUP1 LMID=tpcc GRPNO=1 OPENINFO=NONE

*SERVERS
DEFAULT: CLOPT="-A"
tux_server     SRVGRP=GROUP1 SRVID=1 RQADDR=tpcc REPLYQ=Y
MIN=45 MAX=120

*SERVICES
NEW_ORDER
PAYMENT
ORDER_STATUS
STOCK_LEVEL
```

IIS Registry

```
Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo
Class Name:        <NO CLASS>
Last Write Time:   5/13/98 - 7:41 AM

Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters
Class Name:        <NO CLASS>
Last Write Time:   7/13/98 - 2:34 PM
Value 0
  Name:            BandwidthLevel
  Type:            REG_DWORD
  Data:            0xffffffff

Value 1
  Name:            ListenBackLog
  Type:            REG_DWORD
  Data:            0xc00

Value 2
  Name:            PoolThreadLimit
  Type:            REG_DWORD
  Data:            0x120

Value 3
  Name:            ThreadTimeout
  Type:            REG_DWORD
  Data:            0x15180


Key Name:          SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\Filter
Class Name:        <NO CLASS>
Last Write Time:   5/13/98 - 7:41 AM
Value 0
  Name:            FilterType
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            NumDenySites
  Type:            REG_DWORD
  Data:            0

Value 2
  Name:            NumGrantSites
  Type:            REG_DWORD
  Data:            0


Key Name:
SYSTEM\CurrentControlSet\Services\InetInfo\Parameters\MimeMap
Class Name:        <NO CLASS>
Last Write Time:   5/13/98 - 7:41 AM
Value 0
  Name:            application/envoy,evy,,5
  Type:            REG_SZ
  Data:

Value 1
  Name:            application/mac-binhex40,hqx,,4
  Type:            REG_SZ
  Data:

Value 2
  Name:            application/msword,doc,,5
  Type:            REG_SZ
  Data:

Value 3
  Name:            application/msword,dot,,5
  Type:            REG_SZ
  Data:

Value 4
  Name:            application/octet-stream,*,,5
  Type:            REG_SZ
  Data:

Value 5
  Name:            application/octet-stream,bin,,5
  Type:            REG_SZ
  Data:

Value 6
  Name:            application/octet-stream,exe,,5
  Type:            REG_SZ
  Data:

Value 7
  Name:            application/oda,oda,,5
  Type:            REG_SZ
  Data:

Value 8
  Name:            application/pdf,pdf,,5
  Type:            REG_SZ
  Data:

Value 9
  Name:            application/postscript,ai,,5
  Type:            REG_SZ
  Data:

Value 10
  Name:            application/postscript,eps,,5
  Type:            REG_SZ
  Data:

Value 11
  Name:            application/postscript,ps,,5
  Type:            REG_SZ
  Data:
```

Value 12
Name: application/rtf,rtf,,5
Type: REG_SZ
Data:

Value 13
Name: application/winhlp,hlp,,5
Type: REG_SZ
Data:

Value 14
Name: application/x-bcpio,bcpio,,5
Type: REG_SZ
Data:

Value 15
Name: application/x-cpio,cpio,,5
Type: REG_SZ
Data:

Value 16
Name: application/x-csh,csh,,5
Type: REG_SZ
Data:

Value 17
Name: application/x-director,dcr,,5
Type: REG_SZ
Data:

Value 18
Name: application/x-director,dir,,5
Type: REG_SZ
Data:

Value 19
Name: application/x-director,dxr,,5
Type: REG_SZ
Data:

Value 20
Name: application/x-dvi,dvi,,5
Type: REG_SZ
Data:

Value 21
Name: application/x-gtar,gtar,,9
Type: REG_SZ
Data:

Value 22
Name: application/x-hdf,hdf,,5
Type: REG_SZ
Data:

Value 23
Name: application/x-latex,latex,,5
Type: REG_SZ
Data:

Value 24
Name: application/x-msaccess,mdb,,5
Type: REG_SZ
Data:

Value 25
Name: application/x-mscardfile,crd,,5
Type: REG_SZ
Data:

Value 26
Name: application/x-msclip,clp,,5
Type: REG_SZ
Data:

Value 27
Name: application/x-msexcel,xla,,5
Type: REG_SZ
Data:

Value 28
Name: application/x-msexcel,xlc,,5
Type: REG_SZ
Data:

Value 29
Name: application/x-msexcel,xlm,,5
Type: REG_SZ
Data:

Value 30
Name: application/x-msexcel,xls,,5
Type: REG_SZ
Data:

Value 31
Name: application/x-msexcel,xlt,,5
Type: REG_SZ
Data:

Value 32
Name: application/x-msexcel,xlw,,5
Type: REG_SZ
Data:

Value 33
Name: application/x-msmediaview,m13,,5
Type: REG_SZ
Data:

Value 34
Name: application/x-msmediaview,m14,,5
Type: REG_SZ
Data:

Value 35
Name: application/x-msmetafile,wmf,,5
Type: REG_SZ
Data:

Value 36
Name: application/x-msmoney,mny,,5
Type: REG_SZ
Data:

Value 37
Name: application/x-mspowerpoint,ppt,,5
Type: REG_SZ
Data:

Value 38
Name: application/x-msproject,mpp,,5
Type: REG_SZ
Data:

Value 39
Name: application/x-mspublisher,pub,,5
Type: REG_SZ
Data:

Value 40
Name: application/x-msterminal,trm,,5
Type: REG_SZ
Data:

Value 41
Name: application/x-msworks,wks,,5
Type: REG_SZ
Data:

Value 42
Name: application/x-mswrite,wri,,5
Type: REG_SZ
Data:

Value 43
Name: application/x-netcdf,cdf,,5
Type: REG_SZ
Data:

Value 44
Name: application/x-netcdf,nc,,5
Type: REG_SZ
Data:

Value 45
Name: application/x-perfmon,pma,,5
Type: REG_SZ
Data:

Value 46
Name: application/x-perfmon,pmc,,5
Type: REG_SZ
Data:

Value 47
Name: application/x-perfmon,pml,,5
Type: REG_SZ
Data:

Value 48
Name: application/x-perfmon,pmr,,5
Type: REG_SZ
Data:

Value 49
Name: application/x-perfmon,pmw,,5
Type: REG_SZ
Data:

Value 50
Name: application/x-sh,sh,,5
Type: REG_SZ
Data:

Value 51
Name: application/x-shar,shar,,5
Type: REG_SZ
Data:

Value 52
Name: application/x-sv4cpio,sv4cpio,,5
Type: REG_SZ
Data:

Value 53
Name: application/x-sv4crc,sv4crc,,5
Type: REG_SZ
Data:

Value 54
Name: application/x-tar,tar,,5
Type: REG_SZ

```
Data:

Value 55
  Name:     application/x-tcl,tcl,,5
  Type:     REG_SZ
  Data:

Value 56
  Name:     application/x-tex,tex,,5
  Type:     REG_SZ
  Data:

Value 57
  Name:     application/x-texinfo,texi,,5
  Type:     REG_SZ
  Data:

Value 58
  Name:     application/x-texinfo,texinfo,,5
  Type:     REG_SZ
  Data:

Value 59
  Name:     application/x-troff,roff,,5
  Type:     REG_SZ
  Data:

Value 60
  Name:     application/x-troff,t,,5
  Type:     REG_SZ
  Data:

Value 61
  Name:     application/x-troff,tr,,5
  Type:     REG_SZ
  Data:

Value 62
  Name:     application/x-troff-man,man,,5
  Type:     REG_SZ
  Data:

Value 63
  Name:     application/x-troff-me,me,,5
  Type:     REG_SZ
  Data:

Value 64
  Name:     application/x-troff-ms,ms,,5
  Type:     REG_SZ
  Data:

Value 65
  Name:     application/x-ustar,ustar,,5
  Type:     REG_SZ
  Data:

Value 66
  Name:     application/x-wais-source,src,,7
  Type:     REG_SZ
  Data:

Value 67
  Name:     application/zip,zip,,9
  Type:     REG_SZ
  Data:

Value 68
  Name:     audio/basic,au,,<
  Type:     REG_SZ
  Data:

Value 69
  Name:     audio/basic,snd,,<
  Type:     REG_SZ
  Data:

Value 70
  Name:     audio/x-aiff,aif,,<
  Type:     REG_SZ
  Data:

Value 71
  Name:     audio/x-aiff,aifc,,<
  Type:     REG_SZ
  Data:

Value 72
  Name:     audio/x-aiff,aiff,,<
  Type:     REG_SZ
  Data:

Value 73
  Name:     audio/x-pn-realaudio,ram,,<
  Type:     REG_SZ
  Data:

Value 74
  Name:     audio/x-wav,wav,,<
  Type:     REG_SZ
  Data:

Value 75
  Name:     image/bmp,bmp,,:
  Type:     REG_SZ
  Data:

Value 76
  Name:     image/cis-cod,cod,,5
  Type:     REG_SZ
  Data:

Value 77
  Name:     image/gif,gif,,g
  Type:     REG_SZ
  Data:

Value 78
  Name:     image/ief,ief,,:
  Type:     REG_SZ
  Data:

Value 79
  Name:     image/jpeg,jpe,,:
  Type:     REG_SZ
  Data:

Value 80
  Name:     image/jpeg,jpeg,,:
  Type:     REG_SZ
  Data:

Value 81
  Name:     image/jpeg,jpg,,:
  Type:     REG_SZ
  Data:

Value 82
  Name:     image/tiff,tif,,:
  Type:     REG_SZ
  Data:

Value 83
  Name:     image/tiff,tiff,,:
  Type:     REG_SZ
  Data:

Value 84
  Name:     image/x-cmu-raster,ras,,:
  Type:     REG_SZ
  Data:

Value 85
  Name:     image/x-cmx,cmx,,5
  Type:     REG_SZ
  Data:

Value 86
  Name:     image/x-portable-anymap,pnm,,:
  Type:     REG_SZ
  Data:

Value 87
  Name:     image/x-portable-bitmap,pbm,,:
  Type:     REG_SZ
  Data:

Value 88
  Name:     image/x-portable-graymap,pgm,,:
  Type:     REG_SZ
  Data:

Value 89
  Name:     image/x-portable-pixmap,ppm,,:
  Type:     REG_SZ
  Data:

Value 90
  Name:     image/x-rgb,rgb,,:
  Type:     REG_SZ
  Data:

Value 91
  Name:     image/x-xbitmap,xbm,,:
  Type:     REG_SZ
  Data:

Value 92
  Name:     image/x-xpixmap,xpm,,:
  Type:     REG_SZ
  Data:

Value 93
  Name:     image/x-xwindowdump,xwd,,:
  Type:     REG_SZ
  Data:

Value 94
  Name:     text/html,htm,,h
  Type:     REG_SZ
  Data:

Value 95
  Name:     text/html,html,,h
  Type:     REG_SZ
  Data:

Value 96
  Name:     text/html,stm,,h
  Type:     REG_SZ
  Data:

Value 97
```

```
  Name:         text/plain,bas,,0
  Type:         REG_SZ
  Data:

Value 98
  Name:         text/plain,c,,0
  Type:         REG_SZ
  Data:

Value 99
  Name:         text/plain,h,,0
  Type:         REG_SZ
  Data:

Value 100
  Name:         text/plain,txt,,0
  Type:         REG_SZ
  Data:

Value 101
  Name:         text/richtext,rtx,,0
  Type:         REG_SZ
  Data:

Value 102
  Name:         text/tab-separated-values,tsv,,0
  Type:         REG_SZ
  Data:

Value 103
  Name:         text/x-setext,etx,,0
  Type:         REG_SZ
  Data:

Value 104
  Name:         video/mpeg,mpe,,;
  Type:         REG_SZ
  Data:

Value 105
  Name:         video/mpeg,mpeg,,;
  Type:         REG_SZ
  Data:

Value 106
  Name:         video/mpeg,mpg,,;
  Type:         REG_SZ
  Data:

Value 107
  Name:         video/quicktime,mov,,;
  Type:         REG_SZ
  Data:

Value 108
  Name:         video/quicktime,qt,,;
  Type:         REG_SZ
  Data:

Value 109
  Name:         video/x-msvideo,avi,,<
  Type:         REG_SZ
  Data:

Value 110
  Name:         video/x-sgi-movie,movie,,<
  Type:         REG_SZ
  Data:

Value 111
  Name:         x-world/x-vrml,flr,,5
  Type:         REG_SZ
  Data:

Value 112
  Name:         x-world/x-vrml,wrl,,5
  Type:         REG_SZ
  Data:

Value 113
  Name:         x-world/x-vrml,wrz,,5
  Type:         REG_SZ
  Data:

Value 114
  Name:         x-world/x-vrml,xaf,,5
  Type:         REG_SZ
  Data:

Value 115
  Name:         x-world/x-vrml,xof,,5
  Type:         REG_SZ
  Data:


Key Name:       SYSTEM\CurrentControlSet\Services\InetInfo\Performance
Class Name:     <NO CLASS>
Last Write Time:  5/13/98 - 7:41 AM
Value 0
  Name:         Close
  Type:         REG_SZ
  Data:         CloseINFOPerformanceData

Value 1
  Name:         Collect
  Type:         REG_SZ
  Data:         CollectINFOPerformanceData

Value 2
  Name:         First Counter
  Type:         REG_DWORD
  Data:         0x738

Value 3
  Name:         First Help
  Type:         REG_DWORD
  Data:         0x739

Value 4
  Name:         Last Counter
  Type:         REG_DWORD
  Data:         0x756

Value 5
  Name:         Last Help
  Type:         REG_DWORD
  Data:         0x757

Value 6
  Name:         Library
  Type:         REG_SZ
  Data:         infoctrs.DLL

Value 7
  Name:         Open
  Type:         REG_SZ
  Data:         OpenINFOPerformanceData
```

TPC-C application registry

```
Key Name:       SOFTWARE\Microsoft\TPCC
Class Name:     <NO CLASS>
Last Write Time:  7/14/98 - 8:18 PM
Value 0
  Name:         BackoffDelay
  Type:         REG_SZ
  Data:         500

Value 1
  Name:         ConnectionPooling
  Type:         REG_SZ
  Data:         OFF

Value 2
  Name:         ConnectionPoolRetryTime
  Type:         REG_SZ
  Data:         500

Value 3
  Name:         DeadlockRetry
  Type:         REG_SZ
  Data:         4

Value 4
  Name:         LastInstalledVersion
  Type:         REG_SZ
  Data:         DBLIB

Value 5
  Name:         LOG
  Type:         REG_SZ
  Data:         OFF

Value 6
  Name:         MaxConnections
  Type:         REG_SZ
  Data:         3000

Value 7
  Name:         MaximumWarehouses
  Type:         REG_SZ
  Data:         1600

Value 8
  Name:         NumberOfDeliveryThreads
  Type:         REG_SZ
  Data:         5

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

Value 10
  Name:         QueueSlotts
  Type:         REG_SZ
  Data:         3000
```

WWW Service Registry

```
Key Name:       SYSTEM\CurrentControlSet\Services\W3SVC
Class Name:     <NO CLASS>
Last Write Time:  5/13/98 - 7:41 AM
Value 0
  Name:         DependOnGroup
  Type:         REG_MULTI_SZ
  Data:
```

Value 1
Name: DependOnService
Type: REG_MULTI_SZ
Data: RPCSS
NTLMSSP

Value 2
Name: DisplayName
Type: REG_SZ
Data: World Wide Web Publishing Service

Value 3
Name: ErrorControl
Type: REG_DWORD
Data: 0

Value 4
Name: ImagePath
Type: REG_EXPAND_SZ
Data: C:\WINNT\System32\inetsrv\inetinfo.exe

Value 5
Name: ObjectName
Type: REG_SZ
Data: LocalSystem

Value 6
Name: Start
Type: REG_DWORD
Data: 0x2

Value 7
Name: Type
Type: REG_DWORD
Data: 0x20

Key Name: SYSTEM\CurrentControlSet\Services\W3SVC\Enum
Class Name: <NO CLASS>
Last Write Time: 7/31/98 - 2:24 PM
Value 0
Name: 0
Type: REG_SZ
Data: Root\LEGACY_W3SVC\0000

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

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

Key Name: SYSTEM\CurrentControlSet\Services\W3SVC\Parameters
Class Name: <NO CLASS>
Last Write Time: 5/25/98 - 5:06 PM
Value 0
Name: AcceptExOutstanding
Type: REG_DWORD
Data: 0xc00

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_RTE6

Value 5
Name: Authorization
Type: REG_DWORD
Data: 0x5

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

Value 7
Name: CheckForWAISDB
Type: REG_DWORD
Data: 0

Value 8
Name: ConnectionTimeOut
Type: REG_DWORD
Data: 0x1c20

Value 9
Name: DebugFlags
Type: REG_DWORD
Data: 0x8

Value 10
Name: Default Load File
Type: REG_SZ
Data: Default.htm

Value 11
Name: Dir Browse Control
Type: REG_DWORD
Data: 0x4000001e

Value 12
Name: Filter DLLs
Type: REG_SZ
Data: C:\WINNT\System32\inetsrv\sspifilt.dll

Value 13
Name: GlobalExpire
Type: REG_DWORD
Data: 0xffffffff

Value 14
Name: InstallPath
Type: REG_SZ
Data: C:\WINNT\System32\inetsrv

Value 15
Name: LogFileDirectory
Type: REG_EXPAND_SZ
Data: %SystemRoot%\System32\LogFiles

Value 16
Name: LogFileFormat
Type: REG_DWORD
Data: 0

Value 17
Name: LogFilePeriod
Type: REG_DWORD
Data: 0x1

Value 18
Name: LogFileTruncateSize
Type: REG_DWORD
Data: 0x1388000

Value 19
Name: LogSqlDataSource
Type: REG_SZ
Data: HTTPLOG

Value 20
Name: LogSqlPassword
Type: REG_SZ
Data: sqllog

Value 21
Name: LogSqlTableName
Type: REG_SZ
Data: Internetlog

Value 22
Name: LogSqlUserName
Type: REG_SZ
Data: InternetAdmin

Value 23
Name: LogType
Type: REG_DWORD
Data: 0

Value 24
Name: MajorVersion
Type: REG_DWORD
Data: 0x2

Value 25
Name: MaxConnections
Type: REG_DWORD
Data: 0x186a0

Value 26
Name: MinorVersion
Type: REG_DWORD
Data: 0

Value 27
Name: NTAuthenticationProviders
Type: REG_SZ
Data: NTLM

Value 28
Name: ScriptTimeout
Type: REG_DWORD
Data: 0x384

Value 29
Name: SecurePort
Type: REG_DWORD
Data: 0x1bb

Value 30
Name: ServerSideIncludesEnabled

```
  Type:          REG_DWORD
  Data:          0x1

Value 31
  Name:          ServerSideIncludesExtension
  Type:          REG_SZ
  Data:          .stm


Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Script Map
Class Name:      <NO CLASS>
Last Write Time:  5/13/98 - 7:41 AM
Value 0
  Name:          .idc
  Type:          REG_SZ
  Data:          C:\WINNT\System32\inetsrv\httpodbc.dll


Key Name:
SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots
Class Name:      <NO CLASS>
Last Write Time:  5/14/98 - 6:18 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:  5/13/98 - 7:41 AM
Value 0
  Name:          Close
  Type:          REG_SZ
  Data:          CloseW3PerformanceData

Value 1
  Name:          Collect
  Type:          REG_SZ
  Data:          CollectW3PerformanceData

Value 2
  Name:          First Counter
  Type:          REG_DWORD
  Data:          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:  5/13/98 - 7:41 AM
Value 0
  Name:          Security
  Type:          REG_BINARY
  Data:
00000000  01 00 14 80 c0 00 00 00 - cc 00 00 00 14 00 00 00  ................
00000010  34 00 00 00 02 00 20 00 - 01 00 00 00 02 80 18 00  4..... .........
00000020  ff 01 0f 00 01 01 00 00 - 00 00 00 01 00 00 00 00  ................
00000030  20 02 00 00 02 00 8c 00 - 05 00 00 00 00 00 18 00   ...............
00000040  8d 01 02 00 01 01 00 00 - 00 00 00 01 00 00 00 00  ................
00000050  5c 00 53 00 00 00 1c 00 - fd 01 02 00 01 02 00 00  \.S.............
00000060  00 00 00 05 20 00 00 00 - 23 02 00 00 65 00 72 00  .... ...#...e.r.
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 - 65 00 72 00 00 00 1c 00   ... ...e.r.....
00000090  ff 01 0f 00 01 02 00 00 - 00 00 00 05 20 00 00 00  ............ ...
000000a0  25 02 00 00 65 00 72 00 - 00 00 18 00 fd 01 02 00  %...e.r.........
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:  5/13/98 - 7:42 AM
```

<Microsoft SQL Server 7.0 setting>

Startup Parameters

sqlservr -c -x -t3502

-c	Sart SQL Server independently of the Microsoft Windows NT Service Control Manager.
-x	Disable the keeping of CPU time and cache-hit ration statistics.
-t3502	Prints a message to the log at the beginning and end of each checkpoint.

Microsoft SQL Server Stack Size

The default stack size of Microsoft SQL Server 7.0 was changed using the EDITBIN utility. The EDITBIN utility ships with Microsoft Visual C++ .
The command used was to chage the stack size is:

editbin /stack: 131072 sqlservr.exe.

Microsoft SQL Server 7.0 Configuration Parameters

```
1> 2> 3> 4> 5> 6> 7> 8> 9> 10> 11>
--  File:     VERSION.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Returns SQL Server version string


print " "
select convert(char(30), getdate(),9)
print " "

 ------------------------------
 Jul 31 1998  2:27:53:967PM

(1 row affected)

1> 2> 3>
select @@version

 ----------------------------------------------------------------------------
              ----------------------------------------------------------------------------
              ----------------------------------------------------------------------------
              ---------------------------------
 Microsoft SQL Server  7.00 - 7.00.497 (Intel X86)
              Jun  3 1998 14:18:15
              Cop
              yright (c) 1988-1998 Microsoft Corporation
              Enterprise version on Windo
              ws NT



(1 row affected)
1> 2>
1> 2> 3> 4> 5> 6> 7> 8> 9> 10>
--  File:     CONFIG.SQL
--            Microsoft TPC-C Benchmark Kit Ver. 4.00
--            Copyright Microsoft, 1996
--  Purpose:  Collects SQL Server configuration parameters


print " "
select convert(char(30), getdate(),9)
print " "

 ------------------------------
 Jul 31 1998  2:27:54:763PM

(1 row affected)

1> 2> 3> DBCC execution completed. If DBCC printed error messages, contact your system
administrator.
Configuration option changed. Run the RECONFIGURE statement to install.

sp_configure "show advanced",1
1> 2> reconfigure with override
1> 2> sp_configure
 name
              minimum     maximum     config_value run_value
 ----------------------------------------------------------------------------
              ----------- ----------- ------------ -----------
 affinity mask
                      0  2147483647          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
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 query wait (s)
                0  2147483647        600       600
max server memory (MB)
                4  2147483647       2950      2950
max text repl size (B)
                0  2147483647      65536     65536
max worker threads
               10       1024        235       235
media retention
                0       365          0         0
min memory per query (KB)
              512  2147483647       1024      1024
min server memory (MB)
                0  2147483647       2950      2950
nested triggers
                0         1          1         1
network packet size (B)
             4096      65535       4096      4096
open objects
                0  2147483647          0         0
priority boost
                0         1          1         1
query governor cost limit
                0  2147483647          0         0
recovery interval (min)
                0      32767      32767     32767
remote access
                0         1          0         0
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          1         1
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        260       260
user options
                0      4095          0         0
VLM size (MB)
                0  2147483647          0         0

1>
```

<Disk Array configuration>

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

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

3 Channel - 15 Target  DAC960PJ   #1    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru

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

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

3 Channel - 15 Target  DAC960PJ   #2    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru


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

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

3 Channel - 15 Target  DAC960PJ   #3    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru


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

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

3 Channel - 15 Target  DAC960PJ   #4    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [0:1]  [0:2]  [0:5]  [0:6]  [0:8]
Pack  1 : [2:0]  [2:1]  [2:2]  [2:3]  [2:4]  [2:5]  [2:6]
Pack  2 : [2:8]  [2:9]  [2:10]  [2:11]  [2:12]  [2:13]  [2:14]

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

Number of System Drives = 2

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        52098  MB       0        52098  MB   Write Back
    1        121492  MB       0        121492  MB   Write Back


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

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

3 Channel - 15 Target  DAC960PJ   #5    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru


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

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

3 Channel - 15 Target  DAC960PJ   #6    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1
```

```
Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru


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

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

3 Channel - 15 Target  DAC960PJ   #7    Firmware version 4.03


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

Number of Packs = 1

Pack  0 : [0:0]  [0:1]  [0:2]  [0:5]  [0:6]  [0:8]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        52098  MB       0        52098  MB   Write Back


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

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

3 Channel - 15 Target  DAC960PJ   #8    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru


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

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

3 Channel - 15 Target  DAC960PJ   #7    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru


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

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

3 Channel - 15 Target  DAC960PJ   #8    Firmware version 4.03


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

Number of Packs = 3

Pack  0 : [0:0]  [1:0]  [2:0]  [0:1]  [1:1]  [2:1]  [0:2]  [1:2]
Pack  1 : [2:2]  [0:4]  [1:4]  [2:4]  [0:5]  [1:5]  [2:5]  [0:6]
Pack  2 : [1:6]  [2:6]  [0:8]  [1:8]  [2:8]  [0:9]  [1:9]  [2:9]

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

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        208392  MB       0        208392  MB   Write Thru
```

Appendix D : Space Calculation

180 Day Space

Note : Numbers are in KBytes unless otherwise specified						
Warehouses	1478	**tpmC**	18322.67	**tpmC/W**	12.40	
Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	1,478	168	32	10		210
District	15,200	1,696	72	88		1,856
Item	100,000	9,528	96	221		9,845
New-order	13,680,000	216,288	624		30,400	247,312
History	45,600,000	2,533,408	0		420,534	2,953,942
Orders	45,600,000	1,397,704	772,080		360,174	2,529,958
Customer	45,600,000	33,163,640	2,129,752	811,748		36,105,140
Order-line	456,003,360	28,500,216	71,040		4,742,699	33,313,955
Stock	152,000,000	48,640,008	109,024	1,121,228		49,870,260
Totals		114,462,656	3,082,720	1,933,296	5,553,808	125,032,480
DB File Group	**Count**	**Size**	**Needed**	**Overhead**		**Not Needed**
MSSQL70_misc_fg	8	81,920,000	39,447,651	394,477		42,077,873
MSSQL70_cstock_fg	8	90,112,000	86,835,154	868,352		2,408,495
Totals		172,032,000	126,282,804	1,262,828		44,486,368
Dynamic space	31,555,682	Sum of Data for Order, Order-Line and History (excluding free extents)				
Static space	89,185,817	Data + Index + 5% Space + Overhead - Dynamic space				
Free space	6,804,133	Total Seg. Size - Dynamic Space - Static Space - Not Needed				
Daily growth	6,259,100	(Dynamic space/W * 62.5)* tpmC				
Daily spread	(2,584,517)	Free space - 1.5 * Daily growth (zero if negative)				
180 day (KB)	1,215,823,794	Static space + 180 (daily growth + daily spread)				
180 day (GB)	1159.50	Excludes OS, Paging and RDBMS Logs				
Log size (MB)	50000	Total size of log file				
% Log used	24.76	% of log file used during entire run				
Total N-O Txn	2338483	Total count of N-O transactions during entire run				
Log per N-O txn	5.42	Number of Kbytes per New-Order transaction				
8 Hour Log (GB)	45.47	need double for mirroring				
os, file sys, swap	8.00					
	Disks	**Qty**	**Total**	**Needed**	**Extra**	
Database, Sys	8.47	192	1626.24	1,167.50	467.21	
	8.47	1	8.47			
Mirrored Log	8.47	12	101.64	90.94	10.70	

Appendix E : Price Quotation

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

July 31, 1998

Mr. Martin Strakhovsky
Sr. Product Manager - Servers
NEC Computer Corporation
1414 Massachusetts Avenue
Boxborough, MA 01719-2298

via FAX: (978) 635 - 6888

Dear Marty,

Here is the information you requested regarding US 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 and Tools Group

Microsoft Corporation is an equal opportunity employer.

MYLEX
Leading the World in RAID Technology

DATE: July 24, 1998

COMPANY: NEC Corp.

FAX NO.: (978) 635-6888

ATTN: Mr. Mikio Kanemaki
Server HW Engineering

FROM: **Soogil Stephen Cho**
Vice President, Asia Pacific Sales
Phone: (510) 608-2266 (Direct), Fax: (510) 745-7521
e-mail: stephen@mylex.com

MESSAGE (Total number of pages including cover page: 1)

Mr. Kanemaki,

Mylex is pleased to submit the following quotation to you for the DACPJ product.

	Part Number	Description	Unit Price ($)
1.	DACPJ-3-8E-MYJ	3 channel with 8MB EDO	$ 1,845
2.	DBBPG-MYL	Battery backup	$ 345

Notes:
1. Above prices are based on F.O.B., ex-factory, Fremont, California and firm for 90days.
2. Normal warranty is 3 years, and estimated maintenance cost for additional 2 years is $ 50.
3. Normal delivery lead-time is 30 days ARO.

Should you have any questions or require any additional information, please advise.

Best Regards,

3Com
The Enterprise Middleware Solution

Class 5)				
Tier 4 - Large (more than 8, less than 32 CPUs) and Mainframe Systems (Class 6)	Unlimited	$100,000.00	$15,000.00	$22,000.00
Tier 5 - Massively Parallel	Unlimited	$250,000.00	$37,600.00	$55,000.00